Semi-supervised Structured Prediction Models

DISSERTATION

zur Erlangung des akademischen Grades
doctor rerum naturalium
(Dr. rer. nat.)
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät II
Humboldt-Universität zu Berlin

von
Herr Dipl.-Inf. Ulf Brefeld
geboren am 29.10.1973 in Gronau/Westf.

Präsident der Humboldt-Universität zu Berlin:
Prof. Dr. Dr. h.c. Christoph Markschies

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät II:
Prof. Dr. Wolfgang Coy

Gutachter:

1. Prof. Dr. Tobias Scheffer
2. Prof. Dr. Hans-Dieter Burkhard
3. Prof. Dr. Thorsten Joachims

eingereicht am:	27. Juli 2007
Tag der mündlichen Prüfung:	15. Januar 2008

Abstract

Learning mappings between arbitrary structured input and output variables is a fundamental problem in machine learning. It covers many natural learning tasks and challenges the standard model of learning a mapping from independently drawn instances to a small set of labels. Potential applications include classification with a class taxonomy, named entity recognition, and natural language parsing. In these structured domains, labeled training instances are generally expensive to obtain while unlabeled inputs are readily available and inexpensive.

This thesis deals with semi-supervised learning of discriminative models for structured output variables. The analytical techniques and algorithms of classical semi-supervised learning are lifted to the structured setting. Several approaches based on different assumptions of the data are presented. Co-learning, for instance, maximizes the agreement among multiple hypotheses while transductive approaches rely on an implicit cluster assumption. Furthermore, in the framework of this dissertation, a case study on email batch detection in message streams is presented. The involved tasks exhibit an inherent cluster structure and the presented solution exploits the streaming nature of the data.

The different approaches are developed into semi-supervised structured prediction models and efficient optimization strategies thereof are presented. The novel algorithms generalize state-of-the-art approaches in structural learning such as structural support vector machines. Empirical results show that the semi-supervised algorithms lead to significantly lower error rates than their fully supervised counterparts in many application areas, including multi-class classification, named entity recognition, and natural language parsing.

Keywords:
Learning with structured data, Semi-supervised learning, Kernel machines, Natural language processing

Zusammenfassung

Das Lernen aus strukturierten Eingabe- und Ausgabebeispielen ist die Grundlage für die automatisierte Verarbeitung natürlich auftretender Problemstellungen und eine Herausforderung für das Maschinelle Lernen. Die Einordnung von Objekten in eine Klassentaxonomie, die Eigennamenerkennung und das Parsen natürlicher Sprache sind mögliche Anwendungen. Klassische Verfahren scheitern an der komplexen Natur der Daten, da sie die multiplen Abhängigkeiten und Strukturen nicht erfassen können. Zudem ist die Erhebung von klassifizierten Beispielen in strukturierten Anwendungsgebieten aufwändig und ressourcenintensiv, während unklassifizierte Beispiele günstig und frei verfügbar sind.

Diese Arbeit thematisiert halbüberwachte, diskriminative Vorhersagemodelle für strukturierte Daten. Ausgehend von klassischen halbüberwachten Verfahren werden die zugrundeliegenden analytischen Techniken und Algorithmen auf das Lernen mit strukturierten Variablen übertragen. Die untersuchten Verfahren basieren auf unterschiedlichen Prinzipien und Annahmen, wie zum Beispiel der Konsensmaximierung mehrerer Hypothesen im Lernen aus mehreren Sichten, oder der räumlichen Struktur der Daten im transduktiven Lernen. Desweiteren wird in einer Fallstudie zur Email-Batcherkennung die räumliche Struktur der Daten ausgenutzt und eine Lösung präsentiert, die der sequenziellen Natur der Daten gerecht wird.

Aus den theoretischen Überlegungen werden halbüberwachte, strukturierte Vorhersagemodelle und effiziente Optimierungsstrategien abgeleitet. Die empirische Evaluierung umfasst Klassifikationsprobleme, Eigennamenerkennung und das Parsen natürlicher Sprache. Es zeigt sich, dass die halbüberwachten Methoden in vielen Anwendungen zu signifikant kleineren Fehlerraten führen als vollständig überwachte Baselineverfahren.

Schlagwörter:
Lernen mit strukturierten Daten, Halbüberwachtes Lernen, Kernverfahren, Natürliche Sprachverarbeitung

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Inferring mappings between pairs of input and output variables is one of the oldest problems in machine learning. Numerous instantiations of this general task have been investigated, the most prominent of which are classification, where the output variables are discrete, and univariate regression problems, where the output variables are real numbers.

The classical approach to modeling the underlying target concept is based on features that are extracted from the input variables. Features are supposed to capture characteristics of real-world objects and need to be sufficient for inferring the target concept. In a probabilistic framework, the feature vector is also known as the sufficient statistics. However, the minimal set of features is in general unknown for a given task. In the absence of prior knowledge, as many features as possible are usually extracted and fed into a learning algorithm that assesses their respective discriminatory power. State-of-the-art approaches in machine learning, such as support vector machines, utilize kernel functions to efficiently deal with these rich feature sets.

Supervised machine learning approaches aim to find hypotheses that generalize well on new and unseen input data. However, the problem classes that can be addressed by traditional approaches are limited since the output variables need to be univariate. By contrast, natural learning tasks are rarely univariate but rather complex, and thus render classical machine learning approaches as an inappropriate choice.

In recent years, lifting the algorithmic and analytical techniques of standard supervised learning to the structured setting has been a focus of several strands of research in multiple fields. Probabilistic models such as Markov random fields and stochastic grammars are commonly used to capture sequential, spatial, recursive or relational structure of the output variables. This thesis continues prior research in the field of structured prediction models. We contribute to the field of semi-supervised structural learning by

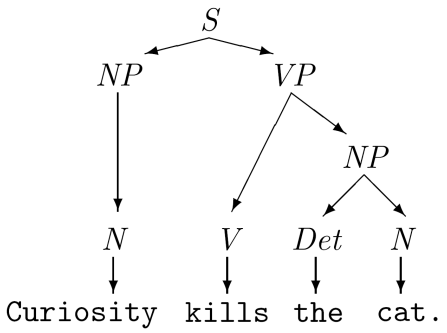


Figure 1.1: *An exemplary parse tree.*

studying settings in which unclassified inputs are given in addition to the labeled sample.

1.1 Structured Learning

Naturally arising learning tasks are complex, highly correlated, and frequently preserve multiple-way dependencies within and between recurrent structures. Exemplary applications of learning with structured input and output variables include named entity recognition and information extraction (sequential output), natural language parsing (tree-structured output, see Figure 1.1), classification with a class taxonomy (here, the output is a node in a tree), and collective classification where the output is a set of interdependent class variables. Learning mappings between those arbitrary structured and interdependent input and output variables challenges the classical model of learning a mapping from independently drawn instances to a small set of labels.

As an introductory example, consider the prediction of protein secondary structures. Proteins are sequences of amino acids and naturally assembled to transcript genetic code to a functional level. Their secondary structure is regarded as the key to their specific three-dimensional shape and thus highly relevant for many fields of computational biology such as artificial protein synthesis. Figure 1.2 shows an example.

For simplicity, we focus in this example only on the two most common secondary structures, *α-helices* and *β-sheets*, and a placeholder, often called *coil*, indicating the absence of the former two. For a protein build of 150 amino acids[1] the cardinality of the set of all possible secondary structures is

[1]The largest known proteins are the *titins* with more than 26,000 amino acids. Yeast

```
KVFGRCELAA AMKRHGLDNY RGYSLGNWVC
 B  HHHHHH HHHHTT TTB TTB HHHHHH

AAKFESNFNT QATNRNTDGS TDYGILQINS
HHHHHHTTBT T EEE TTS  EEETTTTEET

RWWCNDGRTP GSRNLCNIPC SALLSSDITA
TTB B S  T T  BTT SBG GGGGSSS HH

SVNCAKKIVS DGNGMNAWVA WRNRCKGTDV
HHHHHHHHHT SSSGGGGSHH HHHHTTTS G

QAWIRGCRL
GGGTTT
```

Figure 1.2: *The left figure shows the primary (top rows) and secondary structure (bottom rows) of an exemplary protein. The red color indicates α-helices and the green color stands for β-sheets. Spaces within the latter indicate the absence of regular secondary structure. The figure on the right shows the corresponding tertiary structure.*

precisely 3^{150} which is considerably larger than the number of atoms in our galaxy, dark and exotic matter already included. Explicitly enumerating all possible outputs is therefore prohibitive, let alone treating them as learnable class labels.

However, only a small subset of all possible outputs is also likely. Structured prediction models translate this complex learning task into a ranking problem where unlikely sequences are represented only if their incorporation improves the model. Joint feature representations of the input and output variables not only allow for capturing multiple-way dependencies, but have also paved the way to leveraging discriminative learners such as support vector machines to structural problems (Taskar et al., 2004a; Tsochantaridis et al., 2005). Inference in structured models rely on appropriate decoding strategies for the task at hand. Decoding algorithms – such as the Viterbi algorithm for sequential outputs or the CKY algorithm for recursive structures – frequently rely on dynamic programming to compute the top scoring output for a given input.

For instance, the problem of labeling observation sequences has applications that range from language processing tasks such as named entity recognition, part-of-speech tagging, or to biological tasks as already indicated in Figure 1.2. Traditionally, sequence models such as the hidden Markov model for sequential learning and variants thereof have been applied to the label

proteins are on average about 460 amino acids long. Focusing on proteins of a length of 150 is therefore somewhat conservative and serves for exemplary purposes only.

sequence learning problem. Learning procedures for generative models adjust the parameters such that the joint likelihood of training observations and label sequences is maximized. By contrast, from the application point of view, the true benefit of a label sequence predictor corresponds to its ability to find the correct label sequence given an observation sequence.

Empirically, it turns out that discriminative structured prediction models significantly outperform generatively trained models in all application areas. Their success is not only based on discriminative training procedures but also on incorporating arbitrary, possibly rich, feature mappings. Although feature mappings can often be included in generative models, the joint probability highly depends on an accurate estimation of the input distribution; finding an appropriate parameterization is often difficult.

1.2 Semi-supervised Prediction Models for Structured Data

In general, the success of any discriminative model depends on the size of the training sample. Large sample sizes represent the underlying but unknown distribution of examples well and thus allow us to infer models with good generalization properties. However, in structured learning tasks, labeled training pairs are frequently scarce and hard to obtain.

Consider for instance natural language parsing where the task is to predict the most probable parse tree that generates a given input sentence. Parse trees are of fundamental value for further semantic processing of sentences (see also Figure 1.1). To compile an appropriate training set, linguists need to determine the parse trees for all input sentences. While this is absolutely feasible for a few sentences, it becomes quickly tedious when facing thousands of sentences. On the contrary, unlabeled input sentences are abundant and readily available. For instance, sentences in almost any language are electronically accessable, e.g., via the world wide web, at no costs whatsoever. Thus, there is a real need for semi-supervised techniques in structural learning. Emerging questions that will be answered in this thesis include how to integrate the inexpensive, unlabeled instances in the training process of structured models and whether this inclusion is beneficial in terms of predictive performance.

Successful semi-supervised approaches frequently rely on a cluster structure in the data. However, data that does not meet this criteria renders these models inappropriate. In such cases co-learning approaches might be an alternative. Co-learning relies on the existence of multiple views of training

instances and is based on the principle of maximizing the consensus among independent hypotheses trained on these multiple views. We develop these principles into semi-supervised structured prediction models and study their performance empirically. Since the inclusion of unlabeled data is often expensive we devise efficient optimization strategies.

We also study a slightly different variant of semi-supervised learning in a case study on email batch detection. Although, batch detection is an unsupervised task, a ground-truth in form of a labeled training sample exists. Effectively this leads to supervised learning of an unsupervised task and, in essence, makes it an instance of semi-supervised learning. We present a solution based on supervised clustering that exploits the inherent manifold structure in the data and that accounts for the streaming nature of the data.

1.3 Contributions

The central question in this thesis is if unlabeled examples can be used effectively in the training process of discriminative structured prediction models. We therefore continue prior work done in semi-supervised as well as in discriminative structural learning. We leverage semi-supervised techniques to structured domains and obtain novel algorithms that generalize state-of-the-art structured prediction models. The main contributions are as follows.

Co-regularized Least Squares
We introduce the co-learning setting with univariate function approximation. Since the exact solution is cubic in the number of unlabeled instances we propose an approximate variant that scales only linearly with the number of unlabeled instances. We devise appealing closed form solutions for both optimization problems. Moreover, we study a decentralized scenario, where participants keep their labeled data private and agree on a set of unlabeled instances. For the distributed scenario, we devise decentralized optimization strategies for the exact and the approximate regression, where only predictions on unlabeled data need to be shared among the participants. The distributed optimization is also proofed to converge to the global optimum. Empirical results consistently show that the semi-supervised approaches lead to smaller root mean squared errors than fully-supervised baseline methods.

Co-perceptrons and co-support Vector Machines
We lift the consensus maximization principle to the structured domain and derive semi-supervised structured perceptrons for named entity recognition problems. Taking a large margin approach in joint input output space en-

forces confident predictions and we devise co-structural support vector machines (coSVMs) that allow for the direct optimization of arbitrary loss functions. We derive corresponding 1- and 2-norm primal and dual formulations and propose an iterative optimization algorithm that leads to sparse models. Empirically, we show that both semi-supervised approaches outperform their fully-supervised counterpart significantly in multi-class classification, named entity recognition, and natural language parsing tasks.

Transductive Support Vector Machines
We lift the classical inclusion of unlabeled examples by the principle of transduction to the structured domain and devise transductive support vector machines (TSVM) for joint input output spaces. The combinatorial optimization criterion is intractable for arbitrary structured output. We remove the discrete variables by transforming the optimization problem into an unconstrained, and differentiable objective that can be solved efficiently by gradient-based techniques. For the special case of no unlabeled examples, the TSVM resolves to an unconstrained variant of structural support vector machines. Moreover, we derive an unconstrained formulation of coSVMs. Empirically, we observe a significant speed-up in execution time for the unconstrained approaches. We further show that the TSVM can effectively utilize unlabeled data when the implicit cluster assumption on the data is appropriate.

Supervised Clustering of Data Streams
In this case study, we exploit the manifold structure of email batch detection in data streams and translate the problem into a supervised clustering task. We derive a quadratic program whose solution is equivalent to a poly-cut in a fully connected graph and that can be solved in cubic time in the number of email messages. Since cubic time is not tractable for real world applications, we devise a linear time approximation by exploiting the streaming nature of the data. Compared to state-of-the-art methods in supervised clustering, our sequential approach achieves a substantial speed-up in execution time without losing performance. Additional experiments document a reduction of the spam/non-spam misclassification risk by about 40% by using features extracted from the batch information.

1.4 Outline

The outline of this thesis is as follows. We begin with a formal definition of the problem setting in Chapter 2. Chapter 3 then introduces necessary

concepts for semi-supervised learning in structured output spaces. The co-learning setting is introduced with function approximation in Chapter 4 and leveraged to structured output spaces in Chapter 5 with co-perceptrons. We take a large margin approach and present the semi-supervised support vector machine for structured output spaces in Chapter 6. The benefits of continuous optimization are investigated in Chapter 7 where we also derive the transductive support vector machine for structured output variables. We study supervised clustering of data streams with email batch detection in Chapter 8. Chapter 9 provides a conclusion and discusses future work.

1.5 Previously Published Work

Some parts of this thesis have already been published in articles and some of them emerged from collaborations with colleagues. The following list enumerates already published articles and a brief summary of the respective contributions to the papers.

[1] U. Brefeld, C. Büscher and T. Scheffer. Multi-view hidden Markov perceptrons. *Proceedings of the German Workshop on Machine Learning*, 2005.

[2] U. Brefeld, C. Büscher and T. Scheffer. Multi-view discriminative sequential learning, *Proceedings of the European Conference on Machine Learning*, 2005. *Best Paper Award.*

[3] U. Brefeld and T. Scheffer. Semi-supervised learning for structured output variables. *Proceedings of the International Conference on Machine Learning*, 2006.

Papers [1,2] originate from my ideas of lifting co-learning to the structural domain. I developed the theoretical underpinning and the formalism, and planned the experiments that were conducted by Christoph Büscher who also helped with the implementation. I extended the theory of the sequential approach to arbitrary structured output variables on my own in [3]. I also did all the implementing and carried out the experiments.

[4] U. Brefeld, T. Gärtner, T. Scheffer, and S. Wrobel. Efficient co-regularized least squares regression. *Proceedings of the International Conference on Machine Learning*, 2006.

Thomas Gärtner had already started work on co-regression when I joined in. Together, we revised his initial approach and developed the

final formalism and the math together. I implemented the algorithms and conducted the experiments.

[5] P. Haider, U. Brefeld, and T. Scheffer. Discriminative identification of duplicates. *Proceedings of the ECML Workshop on Mining and Learning in Graphs*, 2006.

[6] P. Haider, U. Brefeld, and T. Scheffer. Supervised clustering of streaming data for email batch detection. *Proceedings of the International Conference on Machine Learning*, 2007. *Best Student Paper Award.*

Together with Peter Haider, I realized my ideas on structural batch detection in the papers [5,6]. The theoretical part of either paper was jointly developed by Peter and myself. He implemented the algorithms and carried out the experiments.

[7] A. Zien, U. Brefeld, and T. Scheffer. Transductive support vector machines for structured variables. *Proceedings of the International Conference on Machine Learning*, 2007.

Alex Zien came up with the idea of unconstrained transductive learning for structured ouptut variables. We developed the theory, derived the optimization criterions, implemented the learning algorithms, and conducted the experiments for paper [7] together in equal shares.

Chapter 2

Problem Setting

This thesis deals with semi-supervised learning of structured output variables. Therefore, to establish a sound foundation, we need to introduce generalized concepts of supervised and semi-supervised learning when dealing with structured output spaces.

When the input x and the desired output y are structures, it is not generally feasible to model each possible value of y as an individual class. In addition, not only may there be dependencies between the components of x (e.g., words of a sentence), but also between the components of y (for instance, the class labels of hyperlinked web pages), and between the components of x and y (the semantic annotation of a word may depend on that word, as well as its neighbors). In order to capture these dependencies it is helpful to represent input and output pairs in a joint feature representation. Such joint features of input and output cannot appropriately be modeled when the hypothesis is a function from input to output space; i.e., $f : \mathcal{X} \to \mathcal{Y}$. The learning task is therefore rephrased as finding a function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ such that

$$\hat{y} = \operatorname*{argmax}_{y \in \mathcal{Y}} f(x, y)$$

is the desired output for any input x. Thus, f can be a linear discriminator in a joint space $\Phi(x, y)$ of input and output variables and may depend on arbitrarily defined joint features. Max-margin Markov models (Taskar et al., 2004a), support vector machines for structured output spaces (Tsochantaridis et al., 2005) and other discriminative learners exploit this principle.

We begin with an introduction to supervised machine learning in Section 2.1. The notation of the formal learning task is introduced with respect to later chapters to provide a consistent notation throughout the thesis. The problem setting is then extended to semi-supervised learning in Section 2.2 where we discuss the inclusion of unlabeled examples in the training process.

2.1 Supervised Machine Learning

In this section we will embed classical supervised machine learning into the generalized framework of structured prediction models. This unified view allows us to address structured predictions as well as univariate classification, regression, and ranking tasks at the same time.

In supervised machine learning we are seeking a deterministic mapping from (structured) objects contained in the set $\mathcal{X}$ to an element of the (structured) output or target set $\mathcal{Y}$. The relation between these two sets is assumed by a (in general unknown) joint probability distribution P over domain $\mathcal{X} \times \mathcal{Y}$. Of course, having the joint probability one could use the ratio

$$p(y|x) = \frac{p(x, y)}{p(x)} \tag{2.1}$$

for estimating the probability of output $y \in \mathcal{Y}$ for any given input $x \in \mathcal{X}$. The denominator $p(x)$ is determined by marginalizing the joint probability over all possible target values $y \in \mathcal{Y}$; that is $p(x) = \int_{\mathcal{Y}} p(x, y)dy$.[1] Frequently, the structure of input x also determines the structure of potential outputs y given that input. For instance, in part-of-speech tagging, an input sentence of n tokens gives rise to an output sequence of part-of-speech tags of the same length. Thus, once an input is fixed, $\mathcal{Y}(x)$ denotes the restricted output space induced by input x. For every $x \in \mathcal{X}$ it holds $\mathcal{Y}(x) \in \mathcal{Y}$ and $\mathcal{Y} = \bigcup_{x \in \mathcal{X}} \mathcal{Y}(x)$.

Having the conditional model 2.1 and returning the most likely output $y^\star$ for a given input x suffices for identifying the relation of interest,

$$y^\star = \operatorname*{argmax}_{y \in \mathcal{Y}(x)} p(y|x). \tag{2.2}$$

Equation 2.2 is also known as the maximum a posteriori (MAP) hypothesis. However, in general, the joint probability is unknown and we are given instead a finite sample of n input-output pairs $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$, drawn independently and identically distributed (iid) according to P.

A suitable model for the unknown relation $p(y|x)$ has to account for multiple interactions between inputs and outputs; thus, input examples $x \in \mathcal{X}$ and output examples $y \in \mathcal{Y}$ are represented jointly by a (possibly rich) feature map $\Phi(x, y)$ that allows us to capture these multiple-way dependencies between them. The choice of the joint feature mapping is highly problem dependent and for now we consider it as fixed and appropriately chosen.

The task of finding an appropriate model for $p(y|x)$ can be rephrased in terms of the joint feature representation as choosing a linear discriminant

[1] The integral reduces to a sum over all $y \in \mathcal{Y}$ if $\mathcal{Y}$ is a finite set.

function $f(x, y) = \langle \mathbf{w}, \Phi(x, y) \rangle$ out of a space of candidate functions, the so-called hypothesis space,

$$\mathcal{F} \subseteq \{(x, y) \mapsto \langle \mathbf{w}, \Phi(x, y) \rangle \, : \, \mathbf{w} \in \mathcal{W}\}, \tag{2.3}$$

defined on domain $\mathcal{W}$, such that

$$y_i = \operatorname*{argmax}_{\bar{y} \in \mathcal{Y}} f(x_i, \bar{y}) \tag{2.4}$$

holds for all training pairs $1 \leq i \leq n$. Equation 2.4 says that given an input x_i from the training sample, the highest scoring element of the output space is precisely the corresponding output y_i. We thus call f a decision or ranking function that returns a real value for every argument pair (x_i, y_i) indicating how likely $\bar{y}$ is the correct output for input x_i.

At this point, focusing on generalized linear models may be seen as too restrictive to explain complex relations between inputs and outputs. However, we will see in later chapters that linear models are a natural choice when taking a large margin approach in joint input-output space. Notice that, besides ranking, Equation 2.4 generalizes classical supervised learning tasks. For instance, setting $\mathcal{Y} = \mathbb{R}$, Equation 2.4 leads to a regression setting where the most probable function value at input x is returned; similarly, $\mathcal{Y} = \{c_1, \ldots, c_k\}$ gives rise to classification scenarios, where every instance is assigned one out of k class labels.

We measure the quality of a hypothesis f by a loss function $\ell : \mathcal{Y} \times \mathcal{X} \times \mathcal{F} \to \mathbb{R}$. A simple choice is the 0/1 loss

$$\ell_{0/1}(y, x, f) = [[y \neq \operatorname{argmax}_{\bar{y} \in \mathcal{Y}(x)} f(x, \bar{y})]]$$

that equals 1 if the prediction differs from the true output and is 0 otherwise. From a computational perspective, minimizing the 0/1 loss directly is NP-complete, whereas upper bounds on the 0/1 loss are applied instead. A common choice for classification and ranking tasks is the hinge-loss ℓ that is given by

$$\ell(y, x, f) = \max\{0, 1 + \max_{\substack{\bar{y} \in \mathcal{Y}(x) \\ \bar{y} \neq y}} f(x, \bar{y}) - f(x, y)\}.$$

The hinge loss equals 0 if the true output scores are higher than the top-ranked erroneous prediction plus an additive constant realizing a confidence threshold. Figure 2.1 displays hinge loss and other loss functions upper bounding the 0/1 loss.

Joachims (2005) indicated that the 0/1 loss (or an upper bound) might not be the best choice for a given task since it is independent of the number

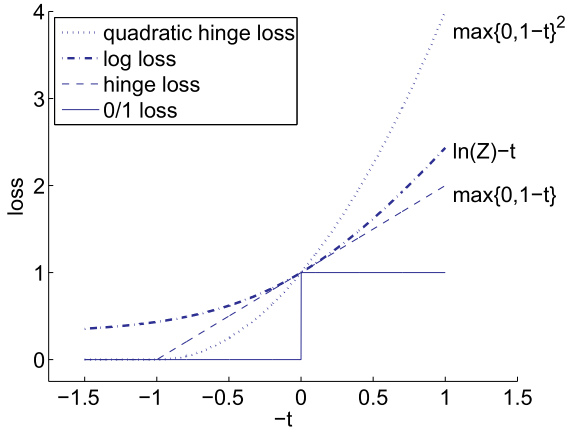


Figure 2.1: *Exemplary loss functions upper bounding the 0/1 loss with* $t = f(x, y) - \max_{\bar{y} \neq y} f(x, \bar{y})$. *Log-loss is shifted to pass through the* $(0, 1)$ *coordinate.*

of mistaken subparts in the output structure. Intuitively, the loss should be small for good guesses, for example predictions that are close to the true target y_i and take large values when the prediction is bogus. We thus introduce an additional task dependent, nonnegative loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_0^+$ that details the distance between the true output y and the prediction $\hat{y} = \operatorname{argmax}_{\bar{y}} f(x_i, \bar{y})$, for instance,

$$\Delta(y_i, \hat{y}) = \begin{cases} 0 & : \quad y_i = \operatorname{argmax}_{\bar{y} \in \mathcal{Y}} f(x_i, \bar{y}) \\ > 0 & : \quad \text{otherwise.} \end{cases}$$

We refer to Δ as structured or task dependent loss. When Δ equals the $0/1$ loss we may utilize hinge loss ℓ directly as an upper bound, see Figure 2.1. For arbitrary choices of Δ, different ways of augmenting the loss functions have been discussed in the literature. The most prominent possibilities are rescaling the confidence threshold of the hinge-loss, (Taskar et al., 2004a)

$$\ell_\Delta^m(y, x, f) = \max\{0, \max_{\substack{\bar{y} \in \mathcal{Y}(x) \\ \bar{y} \neq y}} \{\Delta(y, \bar{y}) + f(x, \bar{y}) - f(x, y)\}\}, \tag{2.5}$$

and rescaling the actual value of the hinge loss (Tsochantaridis et al., 2005),

$$\ell_\Delta^s(y, x, f) = \max\{0, \max_{\substack{\bar{y} \in \mathcal{Y}(x) \\ \bar{y} \neq y}} \{\Delta(y, \bar{y})(1 + f(x, \bar{y}) - f(x, y))\}\}. \tag{2.6}$$

We will address appropriate choices of structured loss functions Δ for exemplary tasks in later chapters. Notice, that for the augmented losses in

Equations 2.5 and 2.6 the relation

$$\ell_\Delta^{m/s}(y, \mathcal{x}, f) \geq \Delta(y, \operatorname{argmax}_{\bar{y} \neq y} f(\mathcal{x}, \bar{y})) \tag{2.7}$$

holds for all $\mathcal{x}$, y, and independently of the choice of Δ. That is, ℓ_Δ^m and ℓ_Δ^s realize upper bounds on the structural loss.

Given an augmented loss function ℓ_Δ, we can restate the optimization problem as finding a function f that realizes the smallest generalization error (the expected risk) given by

$$R(f) = \int_{\mathcal{X} \times \mathcal{Y}} \ell_\Delta(y, \mathcal{x}, f) p(\mathcal{x}, y)\, d\mathcal{x} dy. \tag{2.8}$$

Since the joint probability $p(\mathcal{x}, y)$ is unknown, Equation 2.8 is approximated by its empirical estimate on the iid training sample,

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell_\Delta(y_i, \mathcal{x}_i, f). \tag{2.9}$$

Equation 2.9 is called the empirical risk that converges asymptotically to the expected risk in the limit $n \to \infty$. Similar to directly minimizing the 0/1 loss, minimizing the structured loss directly in Equation 2.9 is also NP-complete for arbitrary Δ.

Initially, minimizing the Estimate 2.9 instead of the true but inaccessible risk may seem to make sense, but this approach suffers from two essential drawbacks. Firstly, the convergence rate of $\hat{R}(f)$ approaching $R(f)$ is slow and large sample sizes are necessary to ensure a small generalization error[2]. Secondly, there is no unique minimum and therefore no distinguished solution since in general there might be several $f \in \mathcal{F}$ realizing $\hat{R}(f) = 0$.

A remedy to ill-posed optimization problems and poor generalization performances is provided by regularization theory. Instead of minimizing Equation 2.9, one minimizes the regularized empirical risk, given by

$$Q(f) = \frac{1}{n} \sum_{i=1}^{n} \ell_\Delta(y_i, \mathcal{x}_i, f) + \eta \|f\|^2, \tag{2.10}$$

where the additive regularization is known as Tikhonov regularization (Tikhonov, 1963). The parameter $\eta > 0$ controls the amount of regularization; that is for $\eta = 0$ we minimize the empirical risk as before, and for $\eta = \infty$ we are seeking smooth functions, irrespectively of the data. Applying Tikhonov regularization is highly related to incorporating priors in a Bayesian formalism (Section A.1).

[2]Actually, with high probability, one can give upper bounds on the expected risk in terms of the empirical risk and a capacity term, (e.g., see Vapnik 1998; Herbrich 2002).

2.2 Semi-supervised Multi-view Learning

The previous section introduced the supervised learning setting as building models from data that realize high predictive performance on unseen instances. The amount of available data is crucial for achieving this goal since the reliability of estimates on the expected error highly depends on the sample size. In the semi-supervised setting, we are not only given n labeled pairs $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$ but also m unlabeled instances $\boldsymbol{x}_{n+1}, \ldots, \boldsymbol{x}_{n+m}$, where every $\boldsymbol{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}(\boldsymbol{x}_i)$. The aim of semi-supervised learning is to utilize these unlabeled data effectively, such that the trained semi-supervised model has a lower generalization error than its fully-supervised counterpart.

Semi-supervised learning (Cooper and Freeman, 1970; Seeger, 2001; Zhu, 2005; Chapelle et al., 2006b) has a long tradition in statistics and machine learning. The expectation maximization (EM) algorithm (Dempster et al., 1977) is probably the most prominent approach to learning from labeled and unlabeled data (McCallum and Nigam, 1998; Nigam and Ghani, 2000). The EM algorithm is wrapped around learning algorithms that fit model parameters to probabilistically labeled data.

Discriminative approaches such as the support vector machine (Boser et al., 1992; Vapnik, 1998) cannot be trained directly from probabilistically labeled data. Several approaches to discriminative semi-supervised learning have been proposed. The transductive SVM (TSVM) (Vapnik, 1998; Bennet and Demiriz, 1998; Joachims, 1999a) still utilizes unlabeled data by EM-like self-labeling and a modification of the optimization criterion. The TSVM is motivated by the idea that the test instances which are to be classified are often available (without class labels) during training. This argues that semi-supervised learning algorithms are applicable in many learning scenarios. Besides transductive SVMs, manifold assumptions of the data (Belkin et al., 2006) or the co-learning strategy (Blum and Mitchell, 1998; Brefeld and Scheffer, 2004) allow the inclusion of unlabeled data in support vector learning.

The underlying assumption in learning manifolds is the existence of high density regions (clusters or manifolds) associated with output values. These approaches are implicitly or explicitly biased to find decision boundaries lying in sparse, low density regions such that two instances are likely to have similar actual output values if they lie within the same cluster or manifold. The postulated, inherent manifold structure of the data is often captured by a directed or undirected graph spanned by the training instances where two examples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are connected with an edge if they are *close* to each other. However, there are many learning tasks, such as function approximation, in which a manifold assumption is an inappropriate choice. Nevertheless, this

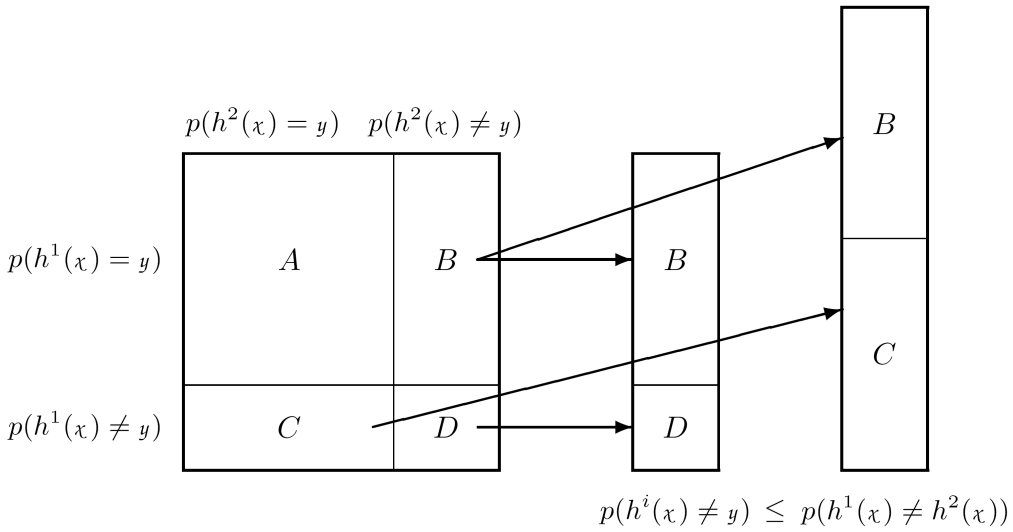


Figure 2.2: *Visualization of the consensus maximization principle for two-view learning with* $h^v(\mathcal{X}) = \text{argmax}_{\bar{y}} f^v(\mathcal{X}, \bar{y})$. *Regions A and D correspond to consensual predictions while areas C and D indicate a disagreement that upper bounds the error rate of either hypothesis.*

is a useful approach that we will explore in greater detail in Chapter 8.

In this thesis, we follow a more general approach of including unlabeled data by taking advantage of the consensus of several hypotheses. The relationship between the consensus of multiple hypotheses and their error rate was first observed by de Sa (1994). She devised a semi-supervised learning method by cascading multi-view vector quantization and linear classification. A multi-view approach to word sense disambiguation combines a classifier that refers to the local context of a word with a second classifier that utilizes the document in which words co-occur (Yarowsky, 1995).

Co-classification (Blum and Mitchell, 1998; Nigam and Ghani, 2000) and co-clustering (Bickel and Scheffer, 2004) are two frameworks for classification and clustering in domains where independent views — i.e., distinct sets of attributes — of labeled and unlabeled data exist. Blum and Mitchell (1998) introduce the co-training algorithm for semi-supervised learning that greedily augments the training sets of two classifiers. Alternatively, a variant of the AdaBoost algorithm has been suggested (Collins and Singer, 1999) that boosts the agreement between two views on unlabeled data. CoEM approaches to semi-supervised learning probabilistically label all unlabeled examples and iteratively exchange those labels between two views (Nigam and Ghani, 2000; Brefeld and Scheffer, 2004). Recently, Meng et al. (2004)

and Farquhar et al. (2006) proposed a fully supervised variant of a co-support vector machine that minimizes the training error as well as the disagreement between two views. Szedmák and Shawe-Taylor (2006) extend this approach to semi-supervised learning and provide generalization bounds.

The intuition behind the co-learning setting is having V independent estimators, judging upon the unknown output of an input example. Dasgupta et al. (2001) studied the relation between the consensus of two independent hypotheses and their error rate. One of their results that holds if the error rate of either hypothesis is smaller than 1/2 is the inequality

$$P\left(\operatorname*{argmax}_{\bar{y}\in\mathcal{Y}(x)} f^1(x,\bar{y}) \neq \operatorname*{argmax}_{\bar{y}'\in\mathcal{Y}(x)} f^2(x,\bar{y}')\right) \geq \max_{v=1,2} P\left(y \neq \operatorname*{argmax}_{\bar{y}\in\mathcal{Y}(x)} f^v(x,\bar{y}))\right).$$

That is, the probability of a disagreement between two independent hypotheses upper bounds the error rate of either hypothesis as indicated in Figure 2.2. Their findings provide a mechanism to minimize the error solely on the basis of unlabeled examples. This insight leads us directly to the consensus maximization principle that is defined as follows.

Definition 2.1 (Consensus maximization principle) *The strategy of semi-supervised multi-view learning can be stated as: minimize the error for labeled examples and maximize the agreement for unlabeled examples.*

Incorporating unlabeled instances according to the consensus maximization principle into Equation 2.10 leads directly to the multi-view objective

$$\begin{aligned} Q_{MV}(\mathbf{f}) =& \frac{1}{nV}\sum_{v=1}^{V}\sum_{i=1}^{n}\ell_\Delta\left(y_i, x_i, f^v\right) + \sum_{v=1}^{V}\eta^v\|f^v\|^2 \\ &+ \frac{\lambda}{mV^2}\sum_{u,v=1}^{V}\sum_{j=n+1}^{n+m}\ell_\Delta\left(\operatorname{argmax}_{\bar{y}} f^u(x_j,\bar{y}), x_j, f^v\right), \end{aligned}$$

that has to be minimized with respect to $\mathbf{f} = (f^1, \ldots, f^V)$. The last term measures the pairwise disagreement between views u and v in terms of the loss ℓ_Δ. The scalar λ acts as a trade-off parameter and determines the overall influence of the disagreement.

However, it should be noted that semi-supervised learning does not *necessarily* lead to better results than supervised learning. When the target distribution is not in the assumed model class, then the best approximation of the unlabeled data can sometimes lie further away from the optimal classifier than the best approximation of (even few) labeled data (Cozman et al., 2003). While additional unlabeled data have often been observed to improve

classifier performance (Baluja, 1998; Collins and Singer, 1999; Nigam et al., 2000; Mladenic, 2002), there are some cases in which they have been found to deteriorate performance – often, but not always, when the labeled sample is large (Shahshahani and Landgrebe, 1994; Baluja, 1998; Nigam et al., 2000; Kockelkorn et al., 2003). Bickel and Scheffer (2007) study cases in which labeled and unlabeled examples are drawn from distinct distributions.

In the co-learning setting that we discuss here, we model V distinct joint input-output mappings of a pair (x, y) which we denote by $\Phi^1(x, y), \Phi^2(x, y), \ldots, \Phi^V(x, y)$. The mappings Φ^v, $1 \leq v \leq V$, are called views, wherefore co-learning is also called multi-view learning. A simple example for 2-view learning is hypertext classification, where we have two natural views on a page, either by the contained text or by the anchor text of its inbound links. In general, there is no straightforward way to split the available attributes into two views. Strategies range from combining hypotheses based on distinct distance metrics (Zhou and Goldman, 2004) to creating additional features for a second view based on distances to pre-computed clusterings (Raskutti et al., 2002). Goldman and Zhou (2000) assume a partitioning of the input space into equivalence classes by the learning algorithms. Surprisingly, in many domains splitting attributes at random into different views and applying a co-classification approach suffices for outperforming single-view learning algorithms (Nigam and Ghani, 2000; Brefeld and Scheffer, 2004). Notice that the representation in each view has to be sufficient for the respective decoding strategy.

In the remainder of this thesis we will frequently consider 2-view learning that is we set $V = 2$, for notational convenience. Note, that all presented approaches are easily generalized to the V-view case with $V \geq 1$. In the 2-view learning scenario we will denote the feature mappings by $\Phi^v(x, y)$, $v = 1, 2$ and write $\Phi^{\bar{v}}$ to indicate the peer view of Φ^v.

Chapter 3

Learning in Joint Input-Output Spaces

In this chapter we detail the prediction of complex and interdepending objects with exemplary learning tasks. We make use of a standard approach in machine learning and utilize graphs to represent dependency structures in underlying multivariate probability models. We exploit input-output structures that can be represented as joint probability models in the exponential family, leading to concise descriptions of the induced joint input-output space.

This chapter is structured as follows. We introduce the exponential family in Section 3.1.1 and identify the involved sufficient statistics as the joint input-output feature mapping that accounts for capturing multiple way dependencies between inputs and outputs. Conditioning the model on the inputs leads directly to conditional random fields that are discussed in Section 3.1.2. Instead of utilizing the derived probability model directly, we take a large margin approach in input-output space and resolve sparse linear models in input-output space in Section 3.1.3, where we also address their co-representation. Exemplary joint feature representations are discussed in greater detail in Section 3.2 for multi-class classification (Section 3.2.1), label sequence learning (Section 3.2.2), and natural language parsing (Section 3.2.3). Each application admits a representation in the exponential family. We also address decoding strategies and appropriate loss functions.

3.1 Graphical Models

In this section we investigate the theoretical background for predicting complex and interdepending variables. We make use of graphical models in the inference machinery that allow us to lift predictions from binary or real values

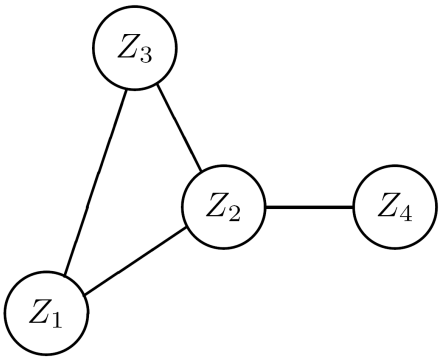


Figure 3.1: *A Markov random field over* $V = \{Z_1, Z_2, Z_3, Z_4\}$.

to complex structures. The dependency structures in multivariate probability models are represented by an underlying graph. Graphs used for the representation may be directed, in which case the model is often referred to as a *Bayesian network* or *belief network*. In case of an undirected graph, the model is called a *Markov random field*. Graphical models have become a broad field; therefore providing a complete survey is beyond the scope of this thesis. We will thus focus on a few relevant models and techniques and direct the interested reader to one of the following excellent books on graphical models (Lauritzen, 1996; Cowell et al., 1999; Jordan and Sejnowski, 2001).

The remainder of this section is organized as follows. Section 3.1.1 introduces Markov random fields that provide the theoretical background for the well known conditional random fields in Section 3.1.2. Section 3.1.3 presents a large margin approach that will be used in the remainder of this thesis.

3.1.1 Markov Random Fields

In Bayesian networks, the underlying directed graph is implicitly given by the causal structure of the problem at hand. Every directed graph can be transformed into an undirected one by connecting parents having the same children. This step is called moralization is and a common preprocessing when dealing with directed graphs. We focus on undirected graphs that are frequently used in areas without such a causal structure such as spatial statistics or natural language processing where dependencies may be bi-directional or even unknown. However, the use of undirected graphs is not restricted to these areas and hybrid models with both directed and undirected edges have also been investigated (Lauritzen, 1996).

We already indicated that the dependency structure of a given problem is encoded by an underlying graph in a one-to-one relation (Pearl, 1988; Lau-

ritzen, 1996). To be more precise, two random variables are connected with an edge if they directly depend on each other. The following definition of conditional independence (Dawid, 1979) quantifies this concept more formally.

Definition 3.1 (Conditional independence) *Given (sets of) random variables A, B, C: We say A is conditionally independent of B given C, and write $A \perp B|C$, if for any valid assignment $B = b$ and $C = c$ the relation $P(A|B = b, C = c) = P(A|C = c)$ holds.*

As an example, consider the set of discrete random variables $V = \{Z_1, \ldots, Z_4\}$ with Z_i taking values in a finite set $\mathcal{Z}_i$. Let $\mathcal{G} = (V, E)$ be an undirected graph over random variables $Z_i \in V$ where $e_{ij} \in E$ draws an edge between Z_i and Z_j as indicated in Figure 3.1. Implying that $\mathcal{G}$ encodes pairwise dependencies between variables V, Definition 3.1 says, knowing the actual value of Z_2, random variable Z_4 is independent of Z_1 and Z_3. We write $Z_4 \perp \{Z_1, Z_3\}|Z_2$ and note that this independence gives rise to the following decomposition of the joint probability as

$$p(V) = p(Z_1, Z_3|Z_2)p(Z_4|Z_2)p(Z_2). \tag{3.1}$$

If such a one-to-one relation between the joint probability and the underlying graph holds, the joint probability distribution is said to obey the *pairwise Markov property* with respect to $\mathcal{G}$ (Pearl, 1988; Lauritzen, 1996). That is, two random variables are unconnected if they are conditionally independent given all other variables,

$$\forall i, j,\ e_{ij} \notin E : Z_i \perp Z_j|V \backslash \{Z_i, Z_j\}. \tag{3.2}$$

More precisely, in these cases we call V a Markov random field (Kemeny et al., 1976) that is defined as follows.

Definition 3.2 (Markov random field) *A collection V of random variables over a finite domain with joint probability P and fulfilling Equation 3.2 with respect to an undirected graph $\mathcal{G}$ is said to be a Markov random field.*

Equation 3.1 already bears one of the key benefits of using Markov random fields. Integrating the prior $p(Z_2)$ into the previous factors shows that the joint probability can equivalently be expressed in terms of the maximal cliques $\mathcal{C} = \{\{2, 4\}, \{1, 2, 3\}\}$ of the underlying graph,

$$p(V) = p(Z_2, Z_4)\, p(Z_1, Z_2, Z_3)/p(Z_2),$$

where the division by $p(Z_2)$ is necessary since Z_2 is contained in both cliques. The fundamental value of this factorization property will soon become clear when we introduce joint feature mappings for exemplary learning tasks by defining features across these maximal cliques. Before that, we need to characterize the joint probability $p(V)$ more precisely by applying a result by Hammersley and Clifford (1971) who link the factorization to V having a Gibbs distribution.

Theorem 3.1 (Hammersley and Clifford) *$V = (Z_1, \ldots, Z_n)$ is a Markov random field with respect to an undirected graph $\mathcal{G} = (V, E)$ if and only if V has a Gibbs distribution (Equation 3.3) with respect to $\mathcal{G}$; that is, the joint probability density function p over V can be written as*

$$p(\mathbf{z}) = Z^{-1} \exp\left\{\sum_{C \in \mathcal{C}} \phi_C(\mathbf{z}_C)\right\}, \tag{3.3}$$

where $\mathbf{z}_C$ denotes the restriction of a valid assignment $\mathbf{z} = (z_1, \ldots, z_n)$ on the maximal cliques $C \in \mathcal{C}$ of $\mathcal{G}$ and ϕ_C denote real-valued functions defined on these maximal cliques. The denominator $Z = \sum_{\mathbf{z}} \exp\{\sum_{C \in \mathcal{C}} \phi_C(\mathbf{z}_C)\}$ is often called the partition function.

Theorem 3.1 says, whenever a set of random variables is also a Markov random field, their joint probability can be written as a Gibbs distribution over the maximal cliques of the dependency graph. The partition function is necessary since the numerator in Equation 3.3 will generally not sum to one for all valid assignments since the only assumption on the potentials is ϕ_C being real valued functions. For the latter reason, we can set $\phi_C(\mathbf{z}_C) = \langle \boldsymbol{\lambda}_C, \phi'_C(\mathbf{z}_C) \rangle$ without restricting the generality; the reason for this inner product decomposition is the interpretation of ϕ'_C as the feature vector of the clique C and $\boldsymbol{\lambda}_C$ as corresponding clique weights. It will be convenient to define the sum over all maximal cliques in $\mathcal{G}$ by

$$\Phi(\mathbf{z}) = \sum_{C \in \mathcal{C}} \phi'_C(\mathbf{z}_C). \tag{3.4}$$

Analogously, summing the $\boldsymbol{\lambda}_C$ over the cliques, $\boldsymbol{\lambda} = \sum_C \boldsymbol{\lambda}_C$, allows us to rewrite Equation 3.3 as a member in the exponential family (Efron, 1978) in its canonical form

$$p(\mathbf{z}|\boldsymbol{\lambda}) = \exp\{\langle \boldsymbol{\lambda}, \Phi(\mathbf{z}) \rangle - g(\boldsymbol{\lambda})\}, \quad \boldsymbol{\lambda} \in \Lambda. \tag{3.5}$$

We call $\Phi(\mathbf{z})$ the sufficient statistics and $\boldsymbol{\lambda}$ the natural parameter. The domain Λ consists of all $\boldsymbol{\lambda}$ having the log-partition function

$$g(\boldsymbol{\lambda}) = \log \sum_{\mathbf{z}} \exp\{\langle \boldsymbol{\lambda}, \Phi(\mathbf{z}) \rangle\} < \infty.$$

The log-partition function is also the moment generating function of the exponential family, that is

$$\frac{\partial}{\partial \boldsymbol{\lambda}} g(\boldsymbol{\lambda}) = \mathbf{E}_{p(\mathbf{z}|\boldsymbol{\lambda})}[\Phi(\mathbf{z})], \quad \frac{\partial^2}{\partial \boldsymbol{\lambda} \partial \boldsymbol{\lambda}} g(\boldsymbol{\lambda}) = \mathbf{Cov}_{p(\mathbf{z}|\boldsymbol{\lambda})}[\Phi(\mathbf{z})], \quad \ldots \tag{3.6}$$

Notice that Equation 3.5 is independent of the number or size of cliques in $\mathcal{G}$ due to the summation over the maximal cliques in Equation 3.4. Exact inference in graphs having a small tree-width can be performed with message passing algorithms, such as the sum-product algorithm (Cowell et al., 1999; Bishop, 2006); the underlying graph is transformed into a junction tree on which messages detailing clique potentials are propagated. However, calculating posterior marginals in arbitrary Markov random fields is known to be NP-hard, that is, exact inference may be intractable when the graph is large and highly connected and one has to resort to approximations such as loopy belief propagation (Pearl, 1988; Murphy et al., 1999).

3.1.2 Conditional Random Fields

The model in Equation 3.5 describes the joint probability of interdependent random variables V, where the dependency structure is determined by an underlying graph $\mathcal{G} = (V, E)$. To derive structured prediction models we decompose random variables contained in V into two disjoint sets X and Y. The variables in X encode a structured input x and the remaining variables Y encode the corresponding output structure y. The underlying graph is required to represent an appropriate dependency structure between elements of $X \cup Y$ for the problem at hand.

Following Equation 3.5, the joint density of an input-output pair (x, y) can be written as a member in the exponential family,

$$p(x, y|\boldsymbol{\lambda}) = \exp\left\{\langle \boldsymbol{\lambda}, \Phi(x, y)\rangle - g(\boldsymbol{\lambda})\right\}. \tag{3.7}$$

Recall that the joint feature mapping $\Phi(x, y)$ of input x and output y is precisely the sufficient statistics in terms of exponential families. A discriminative model predicting the most likely output y for a given x can be derived by conditioning Equation 3.7 on the observation. We obtain a conditional model in the exponential family, also known as a conditional random field (CRF) (Lafferty et al., 2001, 2004),

$$p(y|x; \boldsymbol{\lambda}) = \exp\left\{\langle \boldsymbol{\lambda}, \Phi(x, y)\rangle - g(\boldsymbol{\lambda}|x)\right\},$$

where the log-partition function marginalizes over all possible output values for the given input,

$$g(\boldsymbol{\lambda}|\mathcal{x}) = \log \sum_{y \in \mathcal{Y}(\mathcal{x})} \exp\{\langle \boldsymbol{\lambda}, \Phi(\mathcal{x}, y) \rangle\}.$$

An alternative view on CRFs is provided by the principle of maximum entropy that is frequently used in estimating probability distributions from a training sample containing n input-output pairs $(\mathcal{x}_1, y_1), \ldots, (\mathcal{x}_n, y_n)$. The maximum entropy solution is a distribution that is as uniform as possible and at the same time ensures equality between the expectations with respect to the empirical and the model distribution, respectively (Jaynes, 1957). In terms of CRFs, this becomes apparent when setting the gradient of the log-likelihood to zero to find the optimal parameters $\boldsymbol{\lambda}^\star$. The log-likelihood is given by

$$\log p\left(y_1, \ldots, y_n | \mathcal{x}_1, \ldots, \mathcal{x}_n; \boldsymbol{\lambda}\right) = \sum_{i=1}^{n} \langle \boldsymbol{\lambda}, \Phi(\mathcal{x}_i, y_i) \rangle - g(\boldsymbol{\lambda}|\mathcal{x}_i). \tag{3.8}$$

Differentiating the log-likelihood with respect to the parameter vector $\boldsymbol{\lambda}$ gives (compare also Equation 3.6)

$$\frac{\partial}{\partial \boldsymbol{\lambda}} \log p\left(y_1, \ldots, y_n | \mathcal{x}_1, \ldots, \mathcal{x}_n; \boldsymbol{\lambda}\right) = \mathbf{E}_{\hat{p}(X,Y)}[\Phi(X, Y)] - \sum_{i=1}^{n} \mathbf{E}_{p(Y|\mathcal{x}_i;\boldsymbol{\lambda})}[\Phi(Y, \mathcal{x}_i)],$$

where $\hat{p}$ denotes the empirical distribution of the training data and p is the model distribution. In the optimum both expectations are equal and the solution precisely implements the principle of maximum entropy. Depending on the underlying graph, a closed form solution for the gradient of the log-likelihood is not always achievable. Moreover, maximum likelihood will inevitably lead to bad generalization performance for high-dimensional problems. A remedy can be achieved by placing a prior on the weights, expressing beliefs about parameters before looking at the data.

We are interested in sparse models, having zero weights for redundant and irrelevant features and thus apply a zero mean Gaussian prior with variance σ^2 on the values of $\boldsymbol{\lambda}$, i.e., $\boldsymbol{\lambda} \sim N(\mathbf{0}, \mathbb{1}\sigma^2)$, to exclude degenerate solutions. Following the maximum a posteriori approach and dropping constant terms leads to the posterior distribution of the parameters $\boldsymbol{\lambda}$,

$$\log p(\boldsymbol{\lambda}|\mathcal{D}) \propto \sum_{i=1}^{n} \Big[\langle \boldsymbol{\lambda}, \Phi(\mathcal{x}_i, y_i) \rangle - g(\boldsymbol{\lambda}|\mathcal{x}_i) \Big] - \frac{\boldsymbol{\lambda}^\top \boldsymbol{\lambda}}{2\sigma^2} \tag{3.9}$$

that has to be maximized with respect to $\boldsymbol{\lambda}$. Williams (1999) shows that such a normal prior on the parameters is equivalent to a Gaussian process on $\langle \boldsymbol{\lambda}, \Phi(\mathcal{x}, y) \rangle$ with covariance function $\sigma^2 k(\mathcal{x}, y, \mathcal{x}', y')$, where $k(\mathcal{x}, y, \mathcal{x}', y') = \langle \Phi(\mathcal{x}, y), \Phi(\mathcal{x}', y') \rangle$. From this perspective, one might think of conditional random fields as special cases of Gaussian process classification for structured output spaces (Altun et al., 2004a).

According to Theorem 3.1 and Equation 3.4, the joint feature map decomposes across the cliques of $\mathcal{G}$,

$$\Phi(\mathcal{x}, y) = \sum_{C \in \mathcal{C}} \phi_C(\mathcal{x}_C, y_C).$$

This decomposition devolves to kernels that can also be written in terms of the cliques (Lafferty et al., 2004; Altun et al., 2004b),

$$k\left(\mathcal{x}, y, \mathcal{x}', y'\right) = \sum_{C \in \mathcal{C}} k_C(\mathcal{x}_C, y_C, \mathcal{x}'_C, y'_C).$$

An application of the representer theorem (Wahba, 1990; Schölkopf et al., 2001) shows that the minimizer of the negative log-posterior in Equation 3.9 admits a representation of the form

$$f^\star(\mathcal{x}', y') = \sum_{i=1}^{n} \alpha_i k(\mathcal{x}_i, y_i, \mathcal{x}', y') \tag{3.10}$$

$$= \sum_{c \in \mathcal{C}} \sum_{y_C \in \mathcal{Y}(\mathcal{x}_C)} \bar{\alpha}_C(y_C)\, k_C(\mathcal{x}_C, y_C, \mathcal{x}'_C, y'_C) \tag{3.11}$$

where $(\mathcal{x}_C, y_C)$ denotes the restriction of $(\mathcal{x}, y)$ on the maximal cliques $C \in \mathcal{C}$ of the induced graph $\mathcal{G}$ and the set $\mathcal{Y}(\mathcal{x}_C)$ contains all valid assignments of the clique C. The dual parameters $\alpha_C(y_C)$ detail the importance of an assignment y_C of the clique C. They can be obtained by the equation

$$\bar{\alpha}_C(y_C) = \sum_{i : y_C \in y_i} \alpha_i. \tag{3.12}$$

Marginalizing dual variables over the cliques is effective when the number of different assignments of a clique is small compared to the number of training instances. A similar observation is also used in max-margin Markov networks for obtaining the factored dual representation (Taskar et al., 2004a).

Whether a prior is used or not, that is, whether we maximize Equation 3.8 to derive CRFs or Equation 3.9 to derive kernel CRFs, the optimization is expensive since $\boldsymbol{\lambda}$ has to be adjusted with respect to the complete training sample where the calculation of the partition function is especially time

consuming. Nevertheless, several optimization strategies have been proposed including approaches based on linear programming (Roth and Yih, 2005), iterative scaling (Darroch and Ratcliff, 1972; Lafferty et al., 2001), conjugate gradients (Hestenes and Stiefel, 1952; Sha and Pereira, 2003), Gauss-Newton subspace optimizations (Altun et al., 2004b), gradient tree boosting (Dietterich et al., 2004), and stochastic meta descent (Vishwanathan et al., 2006). Once the optimal parameter vector $\boldsymbol{\lambda}^\star$ is found, predictions for new inputs $\mathcal{x}$ utilize this plug-in estimate $p(\mathcal{y}|\mathcal{x};\boldsymbol{\lambda}^\star)$.

3.1.3 Generalized Linear Models in Multiple Views

An appealing line of research that optimizes a similar criterion as CRFs are large margin approaches for joint input-output spaces, proposed by (Altun et al., 2003b; Taskar et al., 2004a; Tsochantaridis et al., 2005). Starting from the posterior distribution of the parameters $\boldsymbol{\lambda}$, these approaches make explicit use of the argument of the maximum by setting

$$\begin{aligned}\operatorname*{argmax}_{\boldsymbol{\lambda}\in\Lambda} \log p(\boldsymbol{\lambda}|\mathcal{D}) &= \operatorname*{argmax}_{\boldsymbol{\lambda}\in\Lambda} \sum_{i=1}^{n} \Big[\langle \boldsymbol{\lambda}, \Phi(\mathcal{x}_i, \mathcal{y}_i)\rangle - g(\boldsymbol{\lambda}|\mathcal{x}_i)\Big] - \frac{\boldsymbol{\lambda}^\mathsf{T}\boldsymbol{\lambda}}{2\sigma^2} \\ &= \operatorname*{argmin}_{\boldsymbol{\lambda}\in\Lambda} \underbrace{\frac{1}{2}\|\boldsymbol{\lambda}\|^2}_{\textit{regularization term}} + \sigma^2 \underbrace{\sum\nolimits_{i=1}^{n} \Big[g(\mathcal{x}_i;\boldsymbol{\lambda}) - \langle \boldsymbol{\lambda}, \Phi(\mathcal{x}_i, \mathcal{y}_i)\rangle\Big]}_{\textit{empirical risk}}.\end{aligned}$$

We derive the regularized empirical risk of Equation 2.10 with logarithmic loss (log-loss). The variance σ^2 acts as a trade-off parameter between the regularization term and the empirical loss that adjusts the fit to the training data. Thus, starting from graphical models allowing for a representation in the exponential family and taking a large margin approach devolves naturally to regularized risk minimization of generalized linear models in input-output spaces.

For general V-view learning we are essentially looking for V functions from different Hilbert spaces $\mathcal{F}^v$ (possibly defined by different instance descriptions — views — and/or different kernel functions) such that the error of each function on the training data and the disagreement between the functions on the unlabeled data is small. Thus, given n labeled input-output pairs $(\mathcal{x}_1, \mathcal{y}_1), \ldots, (\mathcal{x}_n, \mathcal{y}_n)$ and m unlabeled input examples $\mathcal{x}_{n+1}, \ldots, \mathcal{x}_{n+m}$, we want to find $f^1 : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}, \ldots, f^V : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, i.e., $\mathbf{f} = (f^1, \ldots, f^V) \in$

$\mathcal{F}^1 \times \cdots \times \mathcal{F}^V$ that minimize

$$Q(\mathbf{f}) = \sum_{v=1}^{V} \left[\|f^v\|^2 + C \sum_{i=1}^{n} \ell_\Delta \left(y_i, x_i, f^v \right) \right] + \eta \sum_{u,v=1}^{V} \sum_{j=n+1}^{m} \ell_\Delta(\hat{y}_j^u, x_j, f^v) \tag{3.13}$$

where the norms are in the respective Hilbert spaces and $\hat{y}_j^u = \text{argmax}_{\bar{y}} f^u(x_j, \bar{y})$ denotes the prediction of view u for the j-th unlabeled instance.

A simple application of the *representer theorem* (Wahba, 1990; Schölkopf et al., 2001) in this context shows that the solutions of Equation 3.13 always have the form

$$f_{opt}^v(\cdot, \cdot) = \sum_{i=1}^{n+m} \sum_{\bar{y} \in \mathcal{Y}(x_i)} \alpha_i^v(y_i, \bar{y}) k^v(x_i, y_i, \bar{y}, \cdot, \cdot), \tag{3.14}$$

where

$$k^v(x, y, \bar{y}, x', y') = \langle \Phi^v(x, y), \Phi^v(x', y') \rangle - \langle \Phi^v(x, \bar{y}), \Phi^v(x', y') \rangle$$

is the reproducing kernel of the Hilbert space $\mathcal{F}^v$. In each view v, parameters $\alpha_i^v(y_i, \bar{y})$ weight the relative importance of the pair $(x_i, \bar{y})$ as a negative or pseudo example for the i-th input in the model. As we will see, most of the (exponentially many) α are zero and the model can be stored efficiently. Difference vectors $\Phi(x_i, y_i) - \Phi(x_i, \bar{y})$ associated with non-zero α's are called support vectors.

Equation 3.14 says that we do not have to consider all elements of the reproducing kernel Hilbert space as potential solutions of the regularized risk minimization problem. Instead it is sufficient — without loss of generality — to only consider linear combinations of kernel functions centered at labeled and unlabeled training instances. This can be shown as follows. Any function $f^v \in \mathcal{F}^v$ can be decomposed into a part that lies in the span of the inputs,

$$span \left(\{ \Phi^v(x_i, y) : y \in \mathcal{Y}(x_i),\ i = 1, \ldots, n+m \} \right) \in \mathcal{F}^v \tag{3.15}$$

and a part $f_\perp^v \in \mathcal{F}^v$ perpendicular to it. That is, any $f^v \in \mathcal{F}^v$, $v = 1, 2$, can be written as

$$f^v = \sum_{i=1}^{n+m} \sum_{\bar{y} \in \mathcal{Y}(x_i)} \alpha_i^v(y_i, \bar{y}) (\Phi(x_i, y_i) - \Phi(x_i, \bar{y})) + f_\perp^v, \tag{3.16}$$

with $\langle \Phi^v(x_i, y), f^v_\perp \rangle = 0$ for all $i = 1, \ldots, n+m$ and $y \in \mathcal{Y}(x_i)$. For any labeled or unlabeled input x_j $1 \leq j \leq n+m$, we thus have

$$\begin{aligned} f^v(x_j, y) &= \sum_{i=1}^{n+m} \sum_{\bar{y} \in \mathcal{Y}(x_i)} \alpha_i^v(y_i, \bar{y}) k^v(x, y_i, \bar{y}, x_j, y) + f^v_\perp(x_j, y) \\ &= \sum_{i=1}^{n+m} \sum_{\bar{y} \in \mathcal{Y}(x_i)} \alpha_i^v(y_i, \bar{y}) k^v(x, y_i, \bar{y}, x_j, y) + \sum_{\bar{y}' \in \mathcal{Y}(x_j)} \langle k^v(x_j, y, \bar{y}', \cdot, \cdot), f^v_\perp(\cdot, \cdot) \rangle \\ &= \sum_{i=1}^{n+m} \sum_{\bar{y} \in \mathcal{Y}(x_i)} \alpha_i^v(y_i, \bar{y}) k^v(x, y_i, \bar{y}, x_j, y) \\ &= f^v_{opt}(x_j, y) \end{aligned}$$

using the reproducing property of k^v and orthogonality of k^v and $f^v_\perp$. This shows that the f^v with fixed $\alpha_i(x, \bar{y})$ for all i and $\bar{y}$ form an equivalence class of functions that have the same value at all examples x_i for $1 \leq i \leq n+m$. This can be seen as an analog of the *weak representer theorem*. Now, let us consider the norm of the functions in this equivalence class,

$$\begin{aligned} \left\| f^v \right\|^2 &= \left\| \sum_{i=1}^{n+m} \sum_{\bar{y} \in \mathcal{Y}(x_i)} \alpha_i^v(y_i, \bar{y}) k^v(x, y_i, \bar{y}, \cdot, \cdot) + f^v_\perp \right\|^2 \\ &= \left\| \sum_{i=1}^{n+m} \sum_{\bar{y} \in \mathcal{Y}(x_i)} \alpha_i^v(y_i, \bar{y}) k^v(x, y_i, \bar{y}, \cdot, \cdot) \right\|^2 + \left\| f^v_\perp \right\|^2 \\ &\geq \left\| \sum_{i=1}^{n+m} \sum_{\bar{y} \in \mathcal{Y}(x_i)} \alpha_i^v(y_i, \bar{y}) k^v(x, y_i, \bar{y}, \cdot, \cdot) \right\|^2 \\ &= \left\| f^v_{opt} \right\|^2 . \end{aligned}$$

We observe that for every function $f^v_\perp$ that is not equivalent to the function returning always zero, it holds that $\|f^v\|^2 > \|f^v_{opt}\|^2$. Thus we obtain the analog of the *strong representer theorem*. The significance of this result is that it shows that objectives of the form 3.13 have an optimal solution that can be expressed as kernel expansions in terms of training examples. Corresponding algorithms are thus independent of the dimensionality of the induced feature space and can be optimized efficiently by exploiting the relation

$$\boldsymbol{\lambda}^v = \sum_{i=1}^{n+m} \sum_{\bar{y} \in \mathcal{Y}(x_i)} \alpha_i^v(y_i, \bar{y}) (\Phi^v(x_i, y_i) - \Phi^v(x_i, \bar{y})).$$

Traditionally, the use of kernel functions avoids computing the feature mapping and inner products explicitly. Notice, that in the structured domain, the joint feature mapping depends on the actual values of the output variables; a pre-computation of the kernel is thus prohibitive since we cannot enumerate all potential outputs. We will however see in in the following sections, that inner products in joint feature spaces can frequently be decomposed into an output dependent and an input dependent part. Since the inputs are known beforehand, the latter can be represented by a kernel on the input examples.

Since we draw V hypotheses simultaneously out of V distinct hypothesis spaces, minimizing Equation 3.13 over these multiple views allows us to interpret the disagreement term as an additional data-driven regularization. Consider the case where the hypothesis spaces $\mathcal{F}^v$ for $1 \leq v \leq V$ have an empty intersection and thus do not contain the true labeling function. By minimizing the empirical error and the disagreement we will find hypotheses that are *close* to the true labeling function in each hypothesis space. Thus, similarly to learning under manifold assumptions we have a data driven regularization term that also enforces smooth decisions, but here, *smoothness* is defined with respect to decisions of multiple views. That is, multi-view learning acts as smoothness constraints across hypothesis spaces.

3.2 Joint Feature Representation

So far, we have not yet addressed how to define the joint feature representation $\Phi(\mathcal{x}, y)$ for the task at hand. Connected to a given task, and therefore to the mapping Φ, is not only the decoding strategy but also potential loss functions Δ. In this section, we detail joint feature representations, decoding strategies, and loss functions for four exemplary tasks. To derive a sound methodology, we show that these tasks can be naturally expressed as conditional models in the exponential family. As a consequence, optimization can be performed via generalized linear models by taking a large margin approach in joint input-output space. We begin in Section 3.2.1 with multi-class classification that is extended in Section 3.2.2 to label sequence learning. Section 3.2.3 presents joint feature representations for natural language parsing and Section 3.2.4 introduces the joint model for supervised clustering tasks.

3.2.1 Multi-class Classification

In general, multi-class classification problems can be addressed by a *one-against-one* (Kressel, 1999; Fürnkranz, 2003) or a *one-against-all strategy* (Bouttou et al., 1994; Schölkopf et al., 1995) along with any binary clas-

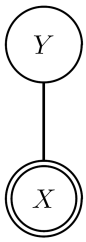


Figure 3.2: *A simple Markov random field for multi-class classification.*

sification algorithm. In the case of k classes, these approaches imply the training of either $k(k-1)/2$ or k different models, respectively, which may be prohibitive in cases of large sample sizes and/or many classes. Multi-class classification can be seen as a special case of learning in joint input-output space where the output space is independent of the input and equals the output alphabet; i.e., $\mathcal{Y} = \mathcal{Y}(\mathcal{x}) = \Sigma$ for all $\mathcal{x} \in \mathcal{X}$ (e.g., see Weston and Watkins 1998; Crammer and Singer 2001). In the remainder of this section, we make use of this identity and rephrase the learning problem as follows.

A natural model for this learning task is the Markov random field from Figure 3.2. According to Section 3.1.1, the conditional density of label y given $\mathcal{x}$ and $\mathbf{w}$ can be expressed as a log-linear combination of the weights $\mathbf{w}$ and the joint feature mapping

$$p(y|\mathcal{x};\mathbf{w}) = \frac{1}{Z(\mathcal{x};\mathbf{w})}\exp\{\langle \mathbf{w}, \Phi(\mathcal{x},y)\rangle\}, \tag{3.17}$$

with the partition function $Z(\mathcal{x};\mathbf{w}) = \sum_{y\in\Sigma}\exp\{\langle \mathbf{w}, \Phi(\mathcal{x},y)\rangle\}$. The joint feature map (Equation 3.18) is given by stacking class-dependent feature vectors for all classes $\sigma_i \in \Sigma$ with $|\Sigma| = k$,

$$\Phi(\mathcal{x},y) = ([[y=\sigma_1]]\psi(\mathcal{x})^\mathsf{T},\ldots,[[y=\sigma_k]]\psi(\mathcal{x})^\mathsf{T})^\mathsf{T}. \tag{3.18}$$

The dimension of the joint feature representation is precisely $|\Sigma|dim(\psi)$. A similar approach is used in (Weston and Watkins, 1998; Crammer and Singer, 2001). The mapping in Equation 3.18 leads to the following inner product in input-output space

$$\langle \Phi(\mathcal{x}_i,y_i), \Phi(\mathcal{x}_j,y_j)\rangle = [[y_i=y_j]]k(\mathcal{x}_i,\mathcal{x}_j),$$

with kernel $k(\mathcal{x}_i,\mathcal{x}_j) = \langle \psi(\mathcal{x}_i), \psi(\mathcal{x}_j)\rangle$. Rewriting Equation 3.17 as a softmax function

$$p(y|\mathcal{x};\mathbf{w}) = \frac{\exp\{\langle \mathbf{w}, \Phi(\mathcal{x},y)\rangle\}}{\sum_{y\in\Sigma}\exp\{\langle \mathbf{w}, \Phi(\mathcal{x},y)\rangle\}}$$

shows that the top scoring label y for a given input $\mathcal{x}$ can be computed via a generalized linear model

$$\hat{y} = \operatorname*{argmax}_{\bar{y}\in\Sigma} p(\bar{y}|\mathcal{x}, \mathbf{w}) = \operatorname*{argmax}_{\bar{y}\in\Sigma} \langle \mathbf{w}, \Phi(\mathcal{x}, \bar{y})\rangle \tag{3.19}$$

because Z is constant and the exponential function is strictly monotonic. Thus, multi-class classification tasks can be captured naturally by the model $f(\mathcal{x}, y) = \langle \mathbf{w}, \Phi(\mathcal{x}, y)\rangle$. Since the number of classes is fixed, we do not need an efficient decoding strategy of Equation 3.19. Instead, we compute $f(\mathcal{x}, \bar{y})$ explicitly for all $\bar{y} \in \Sigma$ and return the highest scoring class.

As in classical multi-class classification settings, class-specific losses can be realized by a *loss matrix* $\mathbf{L} = (\delta_{\tau\sigma})$, where in our case $[\mathbf{L}]_{\tau\sigma} \in \mathbb{R}_0^+$ and $\tau, \sigma \in \Sigma$. Element $[\mathbf{L}]_{\tau\sigma}$, with $\tau \neq \sigma$, denotes the misclassification costs for classifying an object with true class τ erroneously into class σ. Usually, correct classifications imply zero costs, that is, $[\mathbf{L}]_{\tau\tau} = 0$. The corresponding loss function can be stated as $\Delta(y, \hat{y}) = [\mathbf{L}]_{y\hat{y}}$. However, in many tasks, an appropriate loss matrix $\mathbf{L}$ is unknown and the costs for a misclassification are treated independently of the involved class labels. A common choice is setting $[\mathbf{L}]_{\tau\sigma} = 1$ for all $\sigma \neq \tau$; the corresponding loss function reduces to the 0/1-loss

$$\Delta(y, y') = [[y \neq y']].$$

To get a better feeling for our approach, consider the following example where $\Sigma = \{+1, -1\}$. In the binary case $\mathbf{w} = (\mathbf{w}_{-1}^\mathsf{T}, \mathbf{w}_{+1}^\mathsf{T})^\mathsf{T}$ holds and the probability that y takes class $+1$ given an input $\mathbf{x}$ is determined by

$$\begin{aligned} p(y = +1|\mathcal{x}; \mathbf{w}) &= \frac{\exp\{\langle \mathbf{w}, \Phi(\mathcal{x}, +1)\rangle\}}{\exp\{\langle \mathbf{w}, \Phi(\mathcal{x}, -1)\rangle\} + \exp\{\langle \mathbf{w}, \Phi(\mathcal{x}, +1)\rangle\}} \\ &= \left(1 + \exp\left\{\langle \mathbf{w}, \Phi(\mathcal{x}, -1)\rangle - \langle \mathbf{w}, \Phi(\mathcal{x}, +1)\rangle\right\}\right)^{-1} \\ &= \left(1 + \exp\left\{-\left(\langle \mathbf{w}_{+1}, \psi(\mathcal{x})\rangle - \langle \mathbf{w}_{-1}, \psi(\mathcal{x})\rangle\right)\right\}\right)^{-1} \\ &= \left(1 + \exp\left\{-\langle \mathbf{w}_{+1} - \mathbf{w}_{-1}, \psi(\mathcal{x})\rangle\right\}\right)^{-1}. \end{aligned}$$

Defining $\bar{\mathbf{w}} = \mathbf{w}_{+1} - \mathbf{w}_{-1}$, we obtain a *logistic regression* (Equation 3.20) as a special case for binary classification.

$$p(y|\mathcal{x}, \bar{\mathbf{w}}) = \frac{1}{1 + \exp\{-y\langle \bar{\mathbf{w}}, \psi(\mathcal{x})\rangle\}}. \tag{3.20}$$

Due to the implicit log-loss, logistic regression generally leads to non-sparse solutions, see Figure 2.1. According to this section we may also treat logistic regression as a special case of conditional random fields in Section 3.1.2.

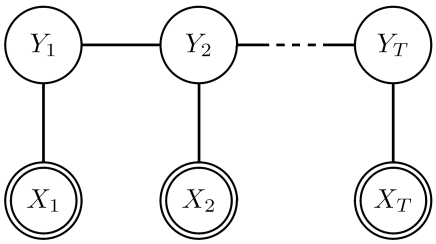


Figure 3.3: *A Markov random field for label sequence learning. The X_i denote observations and the Y_i their corresponding hidden class variables.*

3.2.2 Label Sequence Learning

The problem of labeling observation sequences has applications that range from language processing tasks such as named entity recognition, part-of-speech tagging, and information extraction to biological tasks in which the instances are often DNA strings. Traditionally, sequence models such as the hidden Markov model (Rabiner, 1989; Juang and Rabiner, 1991) and variants thereof have been applied to the label sequence learning problem. Learning procedures for generative models adjust the parameters such that the joint likelihood of training observations and label sequences is maximized. By contrast, from the application point of view, the true benefit of a label sequence predictor corresponds to its ability to find the correct label sequence given an observation sequence. In this section we will derive a conditional model for label sequence learning.

A simple approach to label sequence learning is the use of sliding windows. For every time step t the label y_t is predicted solely on features drawn out of a window of size $2d+1$, that is $x_{t-d}, \ldots, x_t, \ldots, x_{t+d}$, of the neighborhood of x_t but independently of previous labels y_{t-1}. Sliding windows allow the application of any classical learning algorithm such as neural networks (Sejnowski and Rosenberg, 1987) or support vector machines (Hakenberg et al., 2005) and can be extended by recurrent loops to capture the predictions of antecedent time steps $t-d, \ldots, t-1$ as additional inputs for the prediction of time step t (Jordan, 1987; Giles et al., 1994). However, long-range dependencies are rarely captured by window approaches which are thus inappropriate in the absence of prior knowledge. In this section we will derive a conditional model for label sequence learning.

In label sequence learning (Dietterich, 2002) the task is to find a mapping from a sequential input $\boldsymbol{x} = x_1, \ldots, x_T$ to a sequential output $\boldsymbol{y} = y_1, \ldots, y_T$, where $y_i \in \Sigma$; i.e., each element of $\boldsymbol{x}$ is annotated with an element of the output alphabet Σ. We denote the set of all possible labelings of $\boldsymbol{x}$ by $\mathcal{Y}(\boldsymbol{x})$.

When $\mathcal{x}$ is of length T we have $|\mathcal{Y}(\mathcal{x})| = |\Sigma|^T$. If all sequences are of length $T = 1$ we trivially resolve multi-class classification as a special case (see Section 3.2.1).

The label sequence learning task can be modeled in a natural way by a Markov random field shown in Figure 3.3. According to Theorem 3.1, the conditional density of a labeling $\mathcal{y}$ given an observation $\mathcal{x}$ factorizes across the cliques. Similarly to a hidden Markov model, we have dependencies between neighboring labels and between labels and observations. Analogously, decomposing parameters $\mathbf{w}$ into components $\mathbf{w}^A$ and $\mathbf{w}^B$ allows to rewrite the conditional density as

$$p(\mathcal{y}|\mathcal{x};\mathbf{w}) = \frac{1}{Z(\mathcal{x};\mathbf{w})} \prod_{t=2}^{T} \Upsilon^A(y_{t-1}, y_t; \mathbf{w}^A) \prod_{t=1}^{T} \Upsilon^B(x_t, y_t; \mathbf{w}^B), \tag{3.21}$$

where the partition function is given by marginalization over all labelings $\mathcal{y} \in \mathcal{Y}(\mathcal{x})$, that is,

$$Z(\mathcal{x};\mathbf{w}) = \sum_{\mathcal{y}\in\mathcal{Y}(\mathcal{x})} \prod_{t=2}^{T} \Upsilon^A(y_{t-1}, y_t; \mathbf{w}^A) \prod_{t=1}^{T} \Upsilon^B(x_t, y_t; \mathbf{w}^B).$$

We represent the potential functions Υ^A and Υ^B by log-linear combinations of basis functions ϕ^A and ϕ^B that capture *label-label* and *label-observation* interactions, respectively.

$$\Upsilon^A(y_{t-1}, y_t; \mathbf{w}^A) = \exp\left\{\sum_{i=1}^{d_A} w_i^A \phi_i^A(y_{t-1}, y_t)\right\}$$
$$\Upsilon^B(x_t, y_t; \mathbf{w}^B = \exp\left\{\sum_{i=1}^{d_B} w_i^B \phi_i^B(x_t, y_t)\right\}.$$

In many applications, the basis functions are modeled by indicator functions that equal 1 if the corresponding pair exhibits a certain property. For instance, in a part-of-speech tagging task, we might think of a label-label basis function of the form

$$\phi^A_{123}(y_{t-1}, y_t) = [[y_{t-1} = \text{Verb} \wedge\ y_t = \text{Noun}]], \tag{3.22}$$

that equals 1 if a verb is followed by a noun. Analogously, a label-observation basis function may be

$$\phi^B_{234}(x_t, y_t) = [[y_t = \text{Noun} \wedge\ x_t \text{ starts with capital letter}]],$$

where ϕ^B_{234} equals 1 if $\mathcal{x}$ contains a capitalized noun at position t.

Disregarding the normalization constant, the conditional probability in Equation 3.21 can be rewritten in terms of basis functions by

$$\begin{aligned} p(\mathcal{y}|\mathcal{x};\mathbf{w}) &\propto \prod_{t=2}^{T} \Upsilon^A(y_{t-1}, y_t; \mathbf{w}^A) \prod_{t=1}^{T} \Upsilon^B(x_t, y_t; \mathbf{w}^B) \\ &= \prod_{t=2}^{T} \exp\left\{ \sum_{i=1}^{d_A} w_i^A \phi_i^A(y_{t-1}, y_t) \right\} \prod_{t=1}^{T} \exp\left\{ \sum_{i=1}^{d_B} w_i^B \phi_i^B(x_t, y_t) \right\} \\ &= \exp\left\{ \sum_{t=2}^{T} \sum_{i=1}^{d_A} w_i^A \phi_i^A(y_{t-1}, y_t) + \sum_{t=1}^{T} \sum_{i=1}^{d_B} w_i^B \phi_i^B(x_t, y_t) \right\} \\ &= \exp\left\{ \sum_{i=1}^{d_A} w_i^A \sum_{t=2}^{T} \phi_i^A(y_{t-1}, y_t) + \sum_{i=1}^{d_B} w_i^B \sum_{t=1}^{T} \phi_i^B(x_t, y_t) \right\}. \end{aligned}$$

Aggregating the potentials across the cliques leads to

$$\Phi_i^A(\mathcal{x}, \mathcal{y}) = \sum_{t=2}^{T} \phi_i^A(y_{t-1}, y_t) \quad \text{and} \quad \Phi_i^B(\mathcal{x}, \mathcal{y}) = \sum_{t=1}^{T} \phi_i^B(x_t, y_t),$$

where $\Phi_i^{A/B}(\mathcal{x}, \mathcal{y})$ equals the sum of the i-th label-label or label-observation feature, respectively, evaluated along all positions $t = 1, \ldots, T$. We now define the joint feature representation $\Phi(\mathcal{x}, \mathcal{y})$ by stacking up the label-label and the label-observation basis functions

$$\Phi(\mathcal{x}, \mathcal{y}) = (\ldots, \Phi_i^A(\mathcal{x}, \mathcal{y}), \ldots, \Phi_j^B(\mathcal{x}, \mathcal{y}), \ldots)^\mathsf{T}. \tag{3.23}$$

Given the transition features in Equation 3.22, the size of the joint feature mapping is independent of the length of the sequences and determined by $dim(\Phi) = |\Sigma|^2 + |\Sigma| dim(\psi)$. Equation 3.23 together with the weights $\mathbf{w} = (\ldots, w_i^A, \ldots, w_j^B, \ldots)^\mathsf{T}$ show that the conditional probability in Equation 3.21 is precisely a member in the exponential family

$$p(\mathcal{y}|\mathcal{x};\mathbf{w}) = \exp\{\langle \mathbf{w}, \Phi(\mathcal{x}, \mathcal{y}) \rangle - g(\mathcal{x};\mathbf{w})\} \tag{3.24}$$

with partition function $g(\mathcal{x};\mathbf{w}) = \log \sum_{\mathcal{y} \in \mathcal{Y}(\mathcal{x})} \exp\{\langle \mathbf{w}, \Phi(\mathcal{x}, \mathcal{y}) \rangle\} = \log Z(\mathcal{x};\mathbf{w})$. Since we are interested in finding the most likely labeling $\mathcal{y}$ given an observation $\mathcal{x}$, we apply the argmax operator and yield

$$\hat{\mathcal{y}} = \operatorname*{argmax}_{\bar{\mathcal{y}} \in \mathcal{Y}} p(\bar{\mathcal{y}}|\mathcal{x};\mathbf{w}) = \operatorname*{argmax}_{\bar{\mathcal{y}} \in \mathcal{Y}} \langle \mathbf{w}, \Phi(\mathcal{x}, \bar{\mathcal{y}}) \rangle. \tag{3.25}$$

Thus, for our purpose, it suffices to consider generalized linear models of the form

$$f(x, y) = \langle \mathbf{w}, \Phi(x, y) \rangle. \tag{3.26}$$

The described feature map exhibits a first-order Markov property and as a result, decoding can be performed by a *Viterbi algorithm* (Forney, 1973; Schwarz and Chow, 1990) in $\mathcal{O}(T|\Sigma|^2)$ (see Appendix B).

Several loss functions can be applied to sequences to measure the distance between two labelings. Besides the 0/1 loss that only accounts for correctly labeled sequences the Hamming loss is frequently applied to sequence annotation since it allows for the differentiation between slight errors in annotation and sequential trivia. The Hamming loss measures the ratio of incorrect labels per sequence and is given by

$$\Delta_H(y, \hat{y}) = \frac{1}{T} \sum_{t=1}^{T} [[y_t \neq \hat{y}_t]].$$

If all labels $\sigma \in \Sigma$ are equally important, the Hamming loss, detailing the fraction of incorrect tokens, is an appropriate choice. It decomposes over the variables y_t and can thus be applied for instance in Markov networks (Taskar et al., 2004a). However, if there are labels of minor interest, such as the outer tag in named entity recognition, the common Hamming loss is an inappropriate choice. In these cases we might resort to a loss based on the F_1 measure. The F_1 measure is defined as the harmonic average of *precision* and *recall* given by

$$Prec_\sigma(y, \hat{y}) = \frac{\sum_{t=1}^{T} [[y_t = \sigma \wedge \hat{y}_t = \sigma]]}{\sum_{t=1}^{T} [[y'_t = \sigma]]}$$
$$Rec_\sigma(y, \hat{y}) = \frac{\sum_{t=1}^{T} [[y_t = \sigma \wedge \hat{y}_t = \sigma]]}{\sum_{t=1}^{T} [[y_t = \sigma]]}.$$

There are two ways of averaging F_1 scores for multiple labels, the *micro-average* and the *macro-average*. Equation 3.27 details the micro-F_1 score that is averaged over all (relevant) labels $\sigma \in \Sigma$ and consequently dominated by frequent labels.

$$F_1(y, \hat{y}) = \frac{2 \sum_{\sigma \in \Sigma} Prec_\sigma(y, \hat{y}) \sum_{\sigma \in \Sigma} Rec_\sigma(y, \hat{y})}{\sum_{\sigma \in \Sigma} Prec_\sigma(y, \hat{y}) + \sum_{\sigma \in \Sigma} Rec_\sigma(y, \hat{y})} \tag{3.27}$$

Similarly, a macro-F_1 loss can be applied that gives an equal weight on every label, regardless of how rare or common it is. Macro-F_1 scores are therefore

dominated by rare labels (Yang, 1999). In either case, the corresponding loss function is given by

$$\Delta_{F_1}(y, \hat{y}) = 1 - F_1(y, \hat{y}).$$

3.2.3 Natural Language Parsing

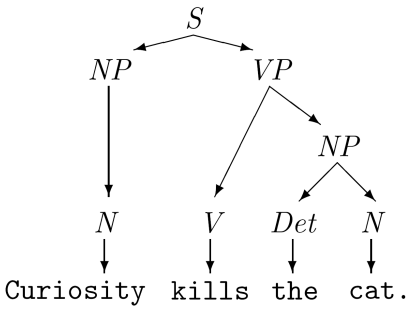
The parse tree of a sentence displays its structure in terms of its dependency structure and is regarded as the primary formalism for further processing such as a semantic analysis. The goal in natural language parsing is therefore to predict the most probable parse tree $y \in \mathcal{Y}(x)$ that generates a given input sentence $x = x_1, \ldots, x_T$ with $x_j \in \Omega$. Each node in the output tree y is generated by a rule of a probabilistic context-free grammar $\mathcal{G}$, a set of recursive production rules used to generate patterns of strings, independently of their context.

Frequently, discriminative natural language parsing models are either based on re-ranking or on dynamic programming. The former generates a set of n-best candidate parse trees and use a classifier to choose one out of them (e.g., Johnson et al. 1999; Collins 2000; Shen et al. 2003). Approaches based on dynamic programming maintain a parse forest or chart to store potential trees (Geman and Johnson, 2002; Clark and Curran, 2004; Miyao and Tsujii, 2002) and output the one with the highest score or probability.

Collins and Duffy (2002) present a parsing approach based on convolutional kernels. Their objective function is optimized with a generalized perceptron algorithm. In this setting, large-margin approaches (Taskar et al., 2004b; Tsochantaridis et al., 2005) can be seen as the best of two worlds. Firstly, in contrast to n-best ranking, they generate only parses that improve the quality of the model (translated into support vectors). Secondly, they apply dynamic programming to decode the top scoring parse tree. Large-margin approaches benefit from this efficient representation and lead to sparse models. To derive conditional models, we first introduce context-free grammars in Definition 3.3 and then extend the concept to probabilistic context-free grammars in Definition 3.4.

Definition 3.3 *A* context-free grammar *(CFG) is defined by a quadruple* $\mathcal{G} = (\mathcal{N}, \Omega, \Sigma, S)$*, where* $\mathcal{N}$ *is the set of* nonterminals*,* Ω *is the set of* terminals*,* $\Omega \cap \mathcal{N} = \emptyset$*,* $\Sigma \subset \mathcal{N} \times (\mathcal{N} \cup \Omega)^*$ *denotes the (finite) set of* rules*, and* $S \in \mathcal{N}$ *is the* start symbol*. Each rule* $A \to \alpha$ *in* Σ *consists of a head* $A \in \mathcal{N}$ *and a body* $\alpha \in (\mathcal{N} \cup \Omega)^*$*.*

A probabilistic context-free grammar generates the same set of parses for a text as the corresponding context-free grammar and assigns a probability to each parse.



$$\Phi(\mathbf{x},\mathbf{y}) = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \begin{array}{lcl} S & \rightarrow & NP, VP \\ NP & \rightarrow & N \\ VP & \rightarrow & V \\ NP & \rightarrow & VP, NP \\ \vdots & & \vdots \\ N & \rightarrow & \texttt{the} \\ N & \rightarrow & \texttt{cat} \end{array}$$

Figure 3.4: *Left: Parse tree for the sentence "`Curiosity kills the cat`". Right: corresponding joint feature map* $\Phi(\mathbf{x},\mathbf{y})$.

Definition 3.4 *The* 5*-tuple* $\mathcal{G} = (\mathcal{N}, \Omega, \Sigma, S, P)$ *is a* probabilistic context-free grammar *(PCFG) if* $(\mathcal{N}, \Omega, \Sigma, S)$ *is a context-free grammar and* $P : \Sigma \rightarrow [0,1]$ *associates a probability* $P(A \rightarrow \alpha)$ *to every production* $(A \rightarrow \alpha) \in \Sigma$ *such that* $\sum_{\alpha \in (\mathcal{N} \cup \Omega)^*} P(A \rightarrow \alpha) = 1$ *for all* $A \in \mathcal{N}$.

The probability of a parse generated by a probabilistic context-free grammar is simply the product of the probabilities of the productions used to generate it. Trivially, every PCFG is also a *weighted context-free grammar* if we identify the weights with the log-probabilities of the productions. We will use the terms probabilistic and weighted context-free grammar interchangeably. In the remainder of this section we assume, for the sake of simplicity, that the grammar $\mathcal{G}$ is in Chomsky normal form[1].

Definition 3.5 *A context-free grammar* $\mathcal{G} = (\mathcal{N}, \Omega, \Sigma, S)$ *is said to be in* Chomsky normal form *if all rules in* Σ *are either of the form* $A \rightarrow a$ *with* $A \in \mathcal{N}$ *and* $a \in \Omega$*, or of the form* $A \rightarrow BC$*, with* $A, B, C \in \mathcal{N}$.

Let $\phi_\sigma(\mathcal{x}, y)$ be the number of times production rule $\sigma \in \Sigma$ occurs in the parse tree y of input sentence $\mathcal{x}$. The conditional probability of y given $\mathcal{x}$ can be written in terms of the counts ϕ_σ as

$$P(y|\mathcal{x}) = \prod_{\sigma \in \Sigma} P(\sigma)^{\phi_\sigma(\mathcal{x}, y)}. \tag{3.28}$$

We define the joint feature representation by stacking up the counts

$$\Phi(\mathcal{x}, y) = (\ldots, \phi_\sigma(\mathcal{x}, y), \ldots)^\mathsf{T}, \tag{3.29}$$

[1]Note that every context-free grammar can always be transformed in a Chomsky normal form.

as indicated in Figure 3.4. By means of 3.29, Equation 3.28 can be equivalently expressed by a Gibbs distribution (Equation 3.3). Introducing weights

$$\mathbf{w} = (\ldots, w_\sigma, \ldots)^\mathsf{T} \quad \text{with} \quad w_\sigma = \ln P(\sigma), \tag{3.30}$$

allows to rewrite the conditional probability of a parse tree in terms of the joint feature representation. We derive

$$\begin{aligned} p(y|x;\mathbf{w}) &= \prod_{\sigma\in\Sigma} P(\sigma)^{\phi_\sigma(x,y)} \\ &= \exp\left\{\langle \mathbf{w}, \Phi(x,y)\rangle\right\} \\ &= \frac{1}{Z(x;\mathbf{w})} \exp\left\{\langle \mathbf{w}, \Phi(x,y)\rangle\right\} \end{aligned} \tag{3.31}$$

since

$$Z(x;\mathbf{w}) = \sum_{y\in\mathcal{Y}} \exp\left\{\langle \mathbf{w}, \Phi(x,y)\rangle\right\} = \sum_{y\in\mathcal{Y}} \prod_{\sigma\in\Sigma} P(\sigma)^{\phi_\sigma(x,y)} = 1.$$

The joint feature map $\Phi(x,y)$ in Equation 3.29 is frequently used (compare Tsochantaridis et al. 2005). However, the general definition of $\Phi(x,y)$ together with the Gibbs distribution allows for a much richer feature map (Chi, 1999). For instance, the mapping $\phi_\sigma(x,y)$ may not only count the number of times production rule σ occurs in the parse y, but it also incorporates structural features, such as the span-length of children nodes (Taskar et al., 2004b).

Equations 3.29 and 3.31 lead to the same generalized linear model as in the previous two sections, independently of the feature map $\Phi(x,y)$. The most probable parse tree y given an input sentence x can be computed by

$$\operatorname*{argmax}_{y\in\mathcal{Y}(x)} p(y|x;\mathbf{w}) = \operatorname*{argmax}_{y\in\mathcal{Y}(x)} \langle \mathbf{w}, \Phi(x,y)\rangle, \tag{3.32}$$

where $\mathcal{Y}(x)$ denotes all valid parse trees with root node S and leaves corresponding to the tokens of x. The weights $\mathbf{w}$ given by Equation 3.30 might not be the best choice. Therefore, we obtain $\mathbf{w}$ by training the generalized model in Equation 3.32. After the learning process, the entries of the weight vector may be interpreted as scores that indicate how likely a certain production rule is to be applied given the local context. Note that the feature map in Equation 3.29 allows for the use of the *Cocke-Kasami-Younger (CKY) algorithm* as an efficient decoding procedure in time $\mathcal{O}(|\Sigma|T^3)$ (Kasami 1965; Younger 1967; see Appendix C). The inner product in input-output space is given by

$$\langle \Phi(x,y), \Phi(x',y')\rangle = \sum_{\sigma\in\Sigma} \phi^\sigma(x,y)\phi^\sigma(x',y').$$

A natural measure for parse trees is the F_1 measure on correct constituents, that is correctly placed triples (A, l, r), where $A \in \mathcal{N}$ is a nonterminal and l and r denote left and right input string positions (Johnson, 1998). We denote the set of all triples of a tree y by $E(y)$ where root nodes and pre-terminal nodes are excluded because they are given as input to the decoder. Thus, the F_1 measure is defined in terms of correct constituents as the harmonic average of *precision* and *recall* given by

$$Prec(y, \hat{y}) = \frac{|E(y) \cap E(\hat{y})|}{|E(\hat{y})|}$$
$$Rec(y, \hat{y}) = \frac{|E(y) \cap E(\hat{y})|}{|E(y)|}.$$

Here, precision equals the fraction of triples in the prediction $\hat{y}$ that also appear in the true y, and recall details the fraction of triples in the true y that are correctly predicted by $\hat{y}$. The corresponding loss function can be stated as

$$\Delta_{F_1}(y, \hat{y}) = 1 - \frac{2Prec(y, \hat{y})Rec(y, \hat{y})}{Prec(y, \hat{y}) + Rec(y, \hat{y})} = 1 - F_1(y, \hat{y}).$$

3.2.4 Supervised Clustering

The task in supervised clustering (Finley and Joachims, 2005) is – like any clustering task – to group similar objects together and dissimilar objects into different groups. But unlike in exploratory data analysis, a *ground truth* of correct clusterings is known beforehand. This allows for learning the similarity function that parameterizes the clustering model such that it correctly groups similar objects in the training data together.

Every input $\mathbf{x} \in \mathcal{X}$ is a collection of T objects, $\mathbf{x} = \{x_1, \ldots, x_T\}$ where every $x_j \in \Omega$. For a training input we are also given the correct clustering as an adjacency matrix $y \in \mathcal{Y}$, with $[y]_{jk} = 1$ if objects x_j and x_k are elements of the same cluster, and 0 otherwise, giving rise to a binary output alphabet $\Sigma = \{0, 1\}$. Due to the symmetry of legal output variables we can restrict output lattices to their lower triangle. The output space $\mathcal{Y}(\mathbf{x})$ for a given input $\mathbf{x}$ therefore consists at most of $2^{(T(T-1)/2)}$ elements.

We capture pairwise similarities of objects by d feature functions $\psi_i : \Omega \times \Omega \to \mathbb{R}$, $1 \leq i \leq d$. The feature functions $\psi_i(x_j, x_k)$ implement aspects of the correspondence between x_j and x_k. For instance, if the discourse area is to cluster email batches we may apply the *tf.idf* similarity of the message bodies, the edit distance of the subject lines, or the similarity of color histograms of images included in the messages. As in the previous sections, it is

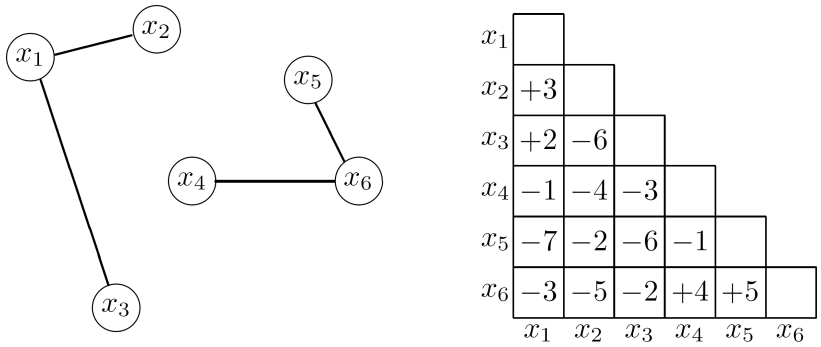


Figure 3.5: *Example detailing correlation clustering. The similarity matrix gives rise to the solution* $\mathcal{C} = \{\{x_1, x_2\}, \{x_3\}, \{x_4, x_5, x_6\}\}$. *Displayed are only edges with positive weights.*

natural to address the problem of learning the similarity measure by linearly combining pairwise feature functions ψ with a weight vector $\mathbf{w}$, forging the parameterized similarity measure

$$sim(x_j, x_k; \mathbf{w}) = \sum_{i=1}^{d} w_i \psi_i(x_j, x_k) = \langle \mathbf{w}, \psi(x_j, x_k) \rangle. \tag{3.33}$$

Applying the similarity function to all pairs of objects in a set $\mathcal{x}_i$ yields a similarity matrix. For a fixed weight vector $\mathbf{w}$, the task is to find the clustering that realizes the maximal inner-cluster similarity summed over all clusters,

$$\hat{y} = \underset{\bar{y} \in \mathcal{Y}(\mathcal{x}_i)}{\operatorname{argmax}} f(\mathcal{x}_i, \bar{y}) = \underset{\bar{y} \in \mathcal{Y}(\mathcal{x}_i)}{\operatorname{argmax}} \sum_{j,k} [\bar{y}]_{jk} \, sim(x_j, x_k). \tag{3.34}$$

Substituting the parameterized counterpart (Equation 3.33) into Equation 3.34 leads directly to the joint feature representation in Equation 3.35. We have

$$\begin{aligned} f(\mathcal{x}, y) &= \sum_{j=1}^{|\mathcal{x}|} \sum_{k=1}^{j-1} [y]_{jk} sim(x_j, x_k) \\ &= \sum_{j=1}^{T} \sum_{k=1}^{j-1} [y]_{jk} \langle \mathbf{w}, \psi(x_j, x_k) \rangle \\ &= \langle \mathbf{w}, \left(\sum_{j=1}^{T} \sum_{k=1}^{j-1} y_{jk} \psi(x_j, x_k) \right) \rangle \\ &= \langle \mathbf{w}, \Phi(\mathcal{x}, y) \rangle, \end{aligned} \tag{3.35}$$

where the joint feature representation can be identified with the sufficient statistics in the exponential family.

Similarly to the previous sections, the learning problem is to find a weight vector $\mathbf{w}$, such that the decoding over the induced similarity matrix yields the correct cluster associations for each object. The problem of creating a consistent clustering of instances from a similarity matrix is equivalent to the problem of *correlation clustering* (Bansal et al. 2002, 2004; for an example see Figure 3.5). In our setting, correlation clustering plays the role of *decoding*; i.e., of generating the clustering $\hat{y}$ for an input set $\mathcal{X}$ when the parameters $\mathbf{w}$ are known. The decoding problem can be stated as

$$\hat{y} = \text{argmax}_{\bar{y} \in \mathcal{Y}(\mathcal{X})} f(\mathcal{X}, \bar{y}) \tag{3.36}$$

$$s.t. \;\; \forall_{jkl} : \quad (1 - [\bar{y}]_{jk}) + (1 - [\bar{y}]_{kl}) \geq (1 - [\bar{y}]_{jl}) \tag{3.37}$$

$$\forall_{jk} \;\; : \quad [\bar{y}]_{jk} \in \{0, 1\}. \tag{3.38}$$

Equation 3.37 is the triangle inequality that requires $\hat{y}$ to be a consistent clustering. It says that if x_j and x_k are elements of the same cluster and x_k and x_l are in the same cluster, then x_j and x_l have to be in the same cluster as well. The obtained solution is equivalent to a poly-cut in a fully connected graph spanned by the objects and their pairwise similarities and theoretically accounts for an infinite number of clusters. However, maximizing $f(\mathcal{X}, y)$ over integer assignments of matrix elements $[y]_{jk}$ is NP-complete. We will exploit continuous approximations of this integer linear program in Chapter 8.

The loss inflicted by a clustering can be quantified by measuring the differences between the true partitioning y of a set and the partitioning predicted by decoding over the similarity matrix, $\hat{y}$. A prominent measure is the Rand index (Rand, 1971) given by

$$Rand(y, \hat{y}) = \frac{2 \sum_{j,k<j} [[\, [y]_{jk} = [\hat{y}]_{jk} \,]]}{T(T-1)}, \tag{3.39}$$

which counts the number of wrong edges in the lower triangle of the adjacency matrix. The corresponding loss Δ is given by

$$\Delta_{Rand}(y, \hat{y}) = 1 - Rand(y, \hat{y}) = \sum_{j,k<j} \frac{2 \, [[\, [y]_{jk} \neq [\hat{y}]_{jk} \,]]}{T(T-1)}.$$

As an alternative, one can also apply a loss that is proportional to the number of objects that are not assigned to their correct cluster. Erroneously assigning an object to a cluster with c elements will incur $2c$ incorrect entries in the cluster matrix. Therefore, we measure the number of incorrectly assigned

objects as

$$\Delta_{total}(y, \hat{y}) = \sum_{j,k:k<j} \frac{| \, [[y]]_{jk} - [\hat{y}]_{jk} \, |}{\sum_{k' \neq j} [y]_{k'k}}.$$

Meila (2003) discusses examples of other potential distance measures.

Chapter 4

Co-regularized Least Squares Regression

In this chapter we examine the co-learning framework with univariate regression models. Despite the increasing popularity of semi-supervised approaches, they have almost exclusively been applied to classification problems until now. We develop the *semi-supervised regression* algorithm co-regularized least squares regression (coRLSR) and propose a semi-parametric variant with improved scalability. CoRLSR is based on casting co-learning as a regularized risk minimization problem in Hilbert spaces. Similar to other kernel methods, the optimal solution in the Hilbert space can be described by a linear combination of kernel functions "centered" on the set of labeled and unlabeled instances. Similar to other semi-supervised approaches, the solution, i.e., the expansion coefficients, can be computed in time cubic in the size of the unlabeled data. As this does not reflect our intuition that semi-supervised learning algorithms should be able to process, and benefit from, huge amounts of unlabeled data, we also develop a semi-parametric approximation that *scales linearly* with the amount of unlabeled data.

We also consider co-regression in a distributed setting. That is, we assume that labeled data is available at different sites and must not be merged (the labels need not be on the same instances and there might be privacy concerns about moving the data). In this setting, we propose a distributed iterative procedure that optimizes the same objective function as for centralized co-regression. Assuming that different views of the same unlabeled data are available at the different sites, the only communication needed in each iteration is to share the predictions of each site about the unlabeled data.

This chapter is structured as follows. We discuss related work in Section 4.1. In Section 4.2 we derive coRLSR and its semi-parametric approximation. The distributed co-regularized least squares regression algorithm is then

presented in Section 4.3. We report our experimental results in Section 4.4 and Section 4.5 provides a conclusion.

4.1 Related Work

Co-classification (Blum and Mitchell, 1998; Nigam and Ghani, 2000) and co-clustering (Bickel and Scheffer, 2004) are two frameworks for classification and clustering in domains where independent views — i.e., distinct sets of attributes — of labeled and unlabeled data exist. Both are based on the observation that the rate of disagreement between independent hypotheses upper bounds their individual error rates (de Sa, 1994). A common application of such approaches is hypertext classification where it can be assumed that the links and text of each web page present two independent views of the same data. However, minimizing the rate of disagreement increases the dependency between the hypotheses and the original motivation for co-learning no longer holds. Nevertheless, the predictive performance of these approaches is often significantly better than for single-view approaches. More surprisingly, in many domains splitting attributes randomly into different views and applying a co-classification approach outperforms single-view learning algorithms (Brefeld and Scheffer, 2004).

De Sa (1994) first observed the relationship between consensus of multiple hypotheses and their error rates and devised a semi-supervised learning method by cascading multi-view vector quantization and linear classification. Blum and Mitchell (1998) introduced the co-training algorithm for semi-supervised learning that greedily augments the training sets of two classifiers. Alternatively, Collins and Singer (1999) suggest a variant of the AdaBoost algorithm that boosts the agreement between two views on unlabeled data.

Dasgupta et al. (2001) and Leskes (2005) give bounds on the error of co-training in terms of the disagreement rate of hypotheses on unlabeled examples in two independent views. This allows the interpretation of the disagreement as an upper bound on the error solely on the basis of unlabeled examples and justifies the direct minimization of the disagreement. The co-EM approach probabilistically labels all unlabeled examples and iteratively exchanges those labels between two views (Nigam and Ghani, 2000). Recently, Farquhar et al. (2006) propose a fully supervised variant of a co-support vector machine that minimizes the training error as well as the disagreement between two views.

Most studies on multi-view and semi-supervised learning consider classification problems, while regression remains largely under-studied. Zhou and Li (2005) apply co-training to kNN regression. Instead of utilizing two disjoint

attribute sets, they use distinct distance measures for the two hypotheses. An approach similar to non-parametric coRLSR has been proposed by Sindhwani et al. (2005a) for classification. Generally, semi-supervised graph-based classification methods can be viewed as function estimators under smoothness constraints, incorporating prior knowledge like smoothness constraints on decision values by regularization techniques (Chung, 1997; Joachims, 2003; Belkin et al., 2004; Zhou et al., 2005; Belkin et al., 2006). These methods utilize kernels on labeled and unlabeled examples. Prominent variants are graph Laplacians (Smola and Kondor, 2003) and RBF or heat kernels (Zhu et al., 2003; Shin et al., 2006), but other similarity matrices are also used (Pozdnoukhov and Bengio, 2006; Verbeek and Vlassis, 2006; Schwaighofer and Tresp, 2003). Krishnapuram et al. (2004) proposed a Bayesian approach to non-transductive graph-based learning.

Transductive approaches to function approximation (Chapelle et al., 1999; Bosnic et al., 2003) have also been extended to Gaussian processes (Seo et al., 2000; Le et al., 2006). Ng et al. (2007) examine semi-supervised regression scenarios when the data contains categorial and continuous variables while Schuurmans et al. (2006) include unlabeled examples by a metric based approach.

4.2 Efficient Co-Regression

In terms of the proposed learning framework, univariate function approximation tasks can be written as a special case with $\mathcal{Y}(\mathcal{x}) = \mathbb{R}$ for all $\mathcal{x}$. Since the output is single-valued there is no need to utilize joint input-output spaces. As a consequence, the hypothesis is independent of an explicit ranking of all output values but implicitly returns the argument of the maximum solely on the basis of the input. We thus obtain the equality

$$\hat{y} = \operatorname*{argmax}_{\bar{y} \in \mathbb{R}} \tilde{f}(\mathcal{x}, y) = f(\mathcal{x}). \tag{4.1}$$

For general V-view learning we are essentially looking for V functions from different Hilbert spaces $\mathcal{F}^v$ (possibly defined by different instance descriptions — views — and/or different kernel functions) such that the error of each function on the training data and the disagreement between the functions on the unlabeled data is small. Note, we are considering a setting slightly more general than most other co-learning approaches: firstly, we directly consider $V \geq 1$ views and secondly, the instances described by different views may differ. We will derive compact matrix representations and use subscripts to denote view indices to omit double superscripts.

We are given V finite sets of training instances $L_v \subseteq \mathcal{X}$, where the projection $\Phi_v(\mathcal{x})$ of instance $\mathcal{x} \in L_v$ onto view v is denoted as d_v-dimensional feature vector $\mathbf{x}_v = (x_1, \ldots, x_{d_v})^\mathsf{T}$. Additionally, we have $\left|\bigcup_{v=1}^{V} L_v\right|$ labels $y(\mathcal{x}) \in \mathbb{R}$, and a set of m instances $U \subseteq \mathcal{X}$ for which the labels are unknown. We will concentrate on squared loss $\ell(y(\mathcal{x}), f_v(\mathcal{x})) = (y(\mathcal{x}) - f_v(\mathcal{x}))^2$ and aim at finding $f_1 : \mathcal{X} \to \mathbb{R}, \ldots, f_V : \mathcal{X} \to \mathbb{R}$, i.e., $\mathbf{f} = (f_1, \ldots, f_V) \in \mathcal{H}_1 \times \cdots \times \mathcal{H}_V$ that minimize

$$Q(\mathbf{f}) = \sum_{v=1}^{V} \left[\sum_{\mathcal{x} \in L_v} (y(\mathcal{x}) - f_v(\mathcal{x}))^2 + \eta \left\| f_v(\cdot) \right\|^2 \right] + \lambda \sum_{u,v=1}^{V} \sum_{\mathcal{x} \in U} (f_u(\mathcal{x}) - f_v(\mathcal{x}))^2 \tag{4.2}$$

where the norms are in the respective Hilbert spaces and λ weights the influence of pairwise disagreements. To avoid cluttering the notation unnecessarily, we omit the obvious generalization of allowing different η for different views. Minimizing Equation 4.2 for $V = 1$ and no unlabeled examples is known as ridge regression (Saunders et al., 1998) or regularized least squares regression (RLSR), see Appendix A for details.

An application of the extended representer theorem (Section 3.1.3) shows that solutions to Equation 4.2 have always the form

$$f_v^{opt} = \sum_{\mathcal{x} \in L_v \cup U} \alpha_v(\mathcal{x}) k_v(\mathcal{x}, \cdot), \tag{4.3}$$

where $k_v(\cdot, \cdot)$ is the reproducing kernel of the Hilbert space $\mathcal{F}_v$. This allows us to express $(f_v(\mathcal{x}_1), f_v(\mathcal{x}_2), \ldots)^\mathsf{T}_{\mathcal{x}_i \in L_v \cup U}$ as $\mathbf{K}_v \boldsymbol{\alpha}_v$ and $\|f_v(\cdot)\|^2$ as $\boldsymbol{\alpha}_v^\mathsf{T} \mathbf{K}_v \boldsymbol{\alpha}_v$, where $[\mathbf{K}_v]_{ij} = \mathbf{K}_v(\mathcal{x}_i, \mathcal{x}_j)$ and $[\boldsymbol{\alpha}_v]_i = \boldsymbol{\alpha}_v(\mathcal{x}_i)$. Here $\mathbf{K}_v$ forms a (strictly) positive definite kernel matrix, i.e., it is symmetric and has no negative (and non zero) eigenvalues. Similarly, we use the notation $\mathbf{y}_v = (y(\mathcal{x}_1), y(\mathcal{x}_2), \ldots)^\mathsf{T}_{\mathcal{x}_i \in L_v}$.

4.2.1 Non-Parametric Least Squares Regression

With n_v training examples in view v and m unlabeled examples, we can rephrase 4.2 and obtain the exact (non-parametric) coRLSR problem:

Optimization Problem 4.1 *Let for each view $v \in \{1, \ldots, V\}$ two matrices $\mathbf{L}_v \in \mathbb{R}^{n_v \times (n_v + m)}$ and $\mathbf{U}_v \in \mathbb{R}^{m \times (n_v + m)}$ be given, such that*

$$\mathbf{K}_v = \begin{pmatrix} \mathbf{L}_v \\ \mathbf{U}_v \end{pmatrix}$$

is strictly positive definite. For fixed $\lambda, \eta \geq 0$ *the* coRLSR optimization problem *is to minimize*

$$Q(\boldsymbol{\alpha}) = \sum_{v=1}^{V} \left[\|\mathbf{y}_v - \mathbf{L}_v \boldsymbol{\alpha}_v\|^2 + \eta \boldsymbol{\alpha}_v^T \mathbf{K}_v \boldsymbol{\alpha}_v \right] + \lambda \sum_{u,v=1}^{V} \|\mathbf{U}_u \boldsymbol{\alpha}_u - \mathbf{U}_v \boldsymbol{\alpha}_v\|^2$$

over $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_V) \in \mathbb{R}^{n_1+m} \times \cdots \times \mathbb{R}^{n_V+m}$.

This optimization problem has been considered in Sindhwani et al. (2005a) for two-view classification. In the remainder of this section we propose a closed form solution and analyze its runtime complexity.

Proposition 4.1 *The solutions* $\boldsymbol{\alpha}_v$ *of the* coRLSR optimization problem *can be found in time* $O(V^3 m^3)$ *(assuming* $m \geq n = \max_v n_v$*).*

Proof With

$$\mathbf{G}_v = \mathbf{L}_v^\mathsf{T} \mathbf{L}_v + \eta \mathbf{K}_v + 2\lambda(V-1)\mathbf{U}_v^\mathsf{T} \mathbf{U}_v$$

we get

$$\nabla_{\boldsymbol{\alpha}_v} Q(\boldsymbol{\alpha}) = 2\mathbf{G}_v \boldsymbol{\alpha}_v - 2\mathbf{L}_v^\mathsf{T} y_v - 4\lambda \sum_{u:u \neq v} \mathbf{U}_v^\mathsf{T} \mathbf{U}_u \boldsymbol{\alpha}_u.$$

At the optimum

$$(\nabla_{\boldsymbol{\alpha}_1} Q(\boldsymbol{\alpha}), \nabla_{\boldsymbol{\alpha}_2} Q(\boldsymbol{\alpha}), \ldots)^\mathsf{T} = \mathbf{0}$$

holds and we can find the exact solution by solving

$$\underbrace{\begin{pmatrix} \mathbf{G}_1 & -2\lambda \mathbf{U}_1^\mathsf{T} \mathbf{U}_2 & \cdots \\ -2\lambda \mathbf{U}_2^\mathsf{T} \mathbf{U}_1 & \mathbf{G}_2 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}}_{=:\mathbf{A}} \begin{pmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} \mathbf{L}_1^\mathsf{T} \mathbf{y}_1 \\ \mathbf{L}_2^\mathsf{T} \mathbf{y}_2 \\ \vdots \end{pmatrix}.$$

The solution requires the inversion of $\mathbf{A}$ which is particularly feasible since $\mathbf{A}$ is strictly positive definite. This can be seen by rewriting $\mathbf{A}$ as the sum $\mathbf{A} = \mathbf{B} + \mathbf{C}$ with

$$\mathbf{B} = \begin{pmatrix} \mathbf{G}_1 - 2\lambda \mathbf{U}_1^\mathsf{T} \mathbf{U}_1 & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{G}_2 - 2\lambda \mathbf{U}_2^\mathsf{T} \mathbf{U}_2 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

is strictly positive definite for $V \geq 2$ and

$$\mathbf{C} = \begin{pmatrix} \lambda \mathbf{U}_1^\mathsf{T} \mathbf{U}_1 & -\lambda \mathbf{U}_1^\mathsf{T} \mathbf{U}_2 & \cdots \\ -\lambda \mathbf{U}_2^\mathsf{T} \mathbf{U}_1 & \lambda \mathbf{U}_2^\mathsf{T} \mathbf{U}_2 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

is also positive definite. Notice for the case $V = 1$ the problem reduces to inverting $\mathbf{G}_1$ which is strictly positive definite as $\mathbf{K}_1$ is strictly positive definite by definition. The solution can thus be found in time $O\left((Vm + Vn)^3\right)$. Using $m > n$ we obtain the bound as stated above. □

For a two-view co-regression the system of equations reduces to

$$\begin{pmatrix} \boldsymbol{\alpha_1} \\ \boldsymbol{\alpha_2} \end{pmatrix} = \begin{pmatrix} \mathbf{G}_1 & -2\lambda \mathbf{U}_1^\mathsf{T}\mathbf{U}_2 \\ -2\lambda \mathbf{U}_2^\mathsf{T}\mathbf{U}_1 & \mathbf{G}_2 \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{L}_1^\mathsf{T}\mathbf{y}_1 \\ \mathbf{L}_2^\mathsf{T}\mathbf{y}_2 \end{pmatrix}$$

and we can use the partitioned inverse equations (Barnett, 1979) to obtain the solution

$$\boldsymbol{\alpha}_1 = \left(\mathbf{G}_1 - 4\lambda^2 \mathbf{U}_1^\mathsf{T}\mathbf{U}_2 {\mathbf{G}_2}^{-1} \mathbf{U}_2^\mathsf{T}\mathbf{U}_1\right)^{-1} \left(\mathbf{L}_1^\mathsf{T}\mathbf{y}_1 + 2\lambda \mathbf{U}_1^\mathsf{T}\mathbf{U}_2 {\mathbf{G}_2}^{-1} \mathbf{L}_2^\mathsf{T}\mathbf{y}_2\right),$$

and correspondingly $\boldsymbol{\alpha}_2$.

4.2.2 Semi-parametric Approximation

While cubic time complexity in the number of labeled examples appears generally acceptable (supervised algorithms like SVMs, RLSR, etc. all have cubic time complexity), cubic time complexity in the number of unlabeled examples renders most real-world problems infeasible as typically $m \gg n$. Still, most state-of-the-art semi-supervised or transductive learning algorithms have cubic or worse time complexity. To achieve lower complexity in the number of unlabeled instances, we resort to a semi-parametric approximation. In particular we optimize over functions that can be expanded in terms of labeled training instances only, that is we approximate the minimizer of Equation 4.2 in each view v by

$$f_v^{opt} = \mathbf{K}_v \boldsymbol{\alpha}_v \approx \mathbf{L}_v \boldsymbol{\alpha}'_v \tag{4.4}$$

where matrices $\mathbf{K}_v$ and $\mathbf{L}_v$ are defined as in Optimization Problem 4.1 with $\boldsymbol{\alpha} \in \mathbb{R}^{n+m}$ and $\boldsymbol{\alpha}' \in \mathbb{R}^n$, respectively. Notice, that the approximation 4.4 violates the extended representer theorem. With n_v training examples in view v and m unlabeled examples, we can phrase the semi-parametric approximation to the coRLSR optimization problem as follows.

Optimization Problem 4.2 *Given for each view $v \in \{1, \ldots, V\}$ a strictly positive definite matrix $\mathbf{L}_v \in \mathbb{R}^{n_v \times n_v}$ and an arbitrary matrix $\mathbf{U}_v \in \mathbb{R}^{m \times n_v}$. For fixed $\lambda, \eta \geq 0$ the* semi-parametric coRLSR optimization problem *is to minimize*

$$Q(\boldsymbol{\alpha}) = \sum_{v=1}^{V} \left[\|\mathbf{y}_v - \mathbf{L}_v \boldsymbol{\alpha}_v\|^2 + \eta \boldsymbol{\alpha}_v^T \mathbf{L}_v \boldsymbol{\alpha}_v\right] + \lambda \sum_{u,v=1}^{V} \|\mathbf{U}_u \boldsymbol{\alpha}_u - \mathbf{U}_v \boldsymbol{\alpha}_v\|^2$$

over $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_V) \in \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_V}$.

Typically, $\mathbf{L}_v$ and $\mathbf{U}_v$ are computed from a strictly positive definite kernel function and form a positive definite kernel matrix $\mathbf{K}_v \in \mathbb{R}^{(n_v+m)\times(n_v+m)}$ as

$$\mathbf{K}_v = \begin{pmatrix} \mathbf{L}_v & \mathbf{U}_v^\mathsf{T} \\ \mathbf{U}_v & * \end{pmatrix}$$

where the part marked by $*$ is not needed.

Proposition 4.2 *The solutions* $\boldsymbol{\alpha}_v$ *of the* semi-parametric coRLSR optimization problem *can be found in time* $O(M^3 n^2 m)$ *(assuming* $m \geq n = \max_v n_v$*).*

Note that the matrices $\mathbf{L}_v$, $\mathbf{U}_v$, and $\mathbf{G}_v$ in the following proof are different from the corresponding matrices in the proof of Theorem 4.1. The symbols are overloaded as they play corresponding roles in either proof. Furthermore, this enables us to prove two theorems at once in the next section.

Proof With

$$\mathbf{G}_v = \mathbf{L}_v^2 + \eta \mathbf{L}_v + 2(V-1)\lambda \mathbf{U}_v^\mathsf{T} \mathbf{U}_v$$

we get

$$\nabla_{\boldsymbol{\alpha}_v} Q(\boldsymbol{\alpha}) = 2\mathbf{G}_v \boldsymbol{\alpha}_v - 2\mathbf{L}_v \mathbf{y}_v - 4\lambda \sum_{u:u\neq v} \mathbf{U}_v^\mathsf{T} \mathbf{U}_u \boldsymbol{\alpha}_u.$$

At the optimum

$$(\nabla_{\boldsymbol{\alpha}_1} Q(\boldsymbol{\alpha}), \nabla_{\boldsymbol{\alpha}_2} Q(\boldsymbol{\alpha}), \ldots)^\mathsf{T} = \mathbf{0}$$

holds and we can find the exact solution by solving

$$\begin{pmatrix} \mathbf{G}_1 & -2\lambda \mathbf{U}_1^\mathsf{T} \mathbf{U}_2 & \cdots \\ -2\lambda \mathbf{U}_2^\mathsf{T} \mathbf{U}_1 & \mathbf{G}_2 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} \mathbf{L}_1 \mathbf{y}_1 \\ \mathbf{L}_2 \mathbf{y}_2 \\ \vdots \end{pmatrix}.$$

Again, this requires the inversion of a strictly positive definite matrix as

$$\begin{pmatrix} G_1 - 2\lambda \mathbf{U}_1^\mathsf{T} \mathbf{U}_1 & \mathbf{0} & \cdots \\ \mathbf{0} & G_2 - 2\lambda \mathbf{U}_2^\mathsf{T} \mathbf{U}_2 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

is strictly positive definite for $M \geq 2$ and

$$\begin{pmatrix} \lambda \mathbf{U}_1^\mathsf{T} \mathbf{U}_1 & -\lambda \mathbf{U}_1^\mathsf{T} \mathbf{U}_2 & \cdots \\ -\lambda \mathbf{U}_2^\mathsf{T} \mathbf{U}_1 & \lambda \mathbf{U}_2^\mathsf{T} \mathbf{U}_2 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

is positive definite. The solution can thus be found in time $O\left((Vn)^3 + V^2 m\right)$. Using $m > n$ we obtain the bound as stated above. □

For the two-view co-regression we can again make use of the partitioned inverse equations to obtain

$$\boldsymbol{\alpha}_1 = \left(\mathbf{G}_1 - 4\lambda^2 \mathbf{U}_1^\mathsf{T}\mathbf{U}_2{\mathbf{G}_2}^{-1}\mathbf{U}_2^\mathsf{T}\mathbf{U}_1\right)^{-1}\left(\mathbf{L}_1\mathbf{y}_1 + 2\lambda\mathbf{U}_1^\mathsf{T}\mathbf{U}_2{\mathbf{G}_2}^{-1}\mathbf{L}_2\mathbf{y}_2\right).$$

and correspondingly $\boldsymbol{\alpha}_2$.

4.2.3 Relation to RLSR

It turns out that the above two Optimization Problems 4.1 and 4.2 are natural generalizations of regularized least squares regression. In both cases for $m = 0$ we obtain V independent regularized least squares solutions. In the semi-parametric case we also obtain V independent regularized least squares solutions for $\lambda = 0$. The V solutions can be combined to a single model by averaging their predictions. For $V = 1$ the agreement term (the second part of the objective function in Optimization Problem 4.2) disappears and we recover a single regularized least squares solution. In the non-parametric case for $\lambda = 0$ or $V = 1$ the optimization problem still appears different from the regularized least squares optimization problem as the regularization term for each view includes a regularization over the unlabeled data. However, applying the representer theorem to this case shows immediately that all components of $\boldsymbol{\alpha}_v$ corresponding to unlabeled data will be zero for the minimizer of the optimization problem. This shows that non-parametric as well as semi-parametric coRLSR contain traditional RLSR as a special case and can therefore both be seen as natural generalizations.

4.3 Distributed CoRLSR

Machine learning traditionally considers application scenarios where the data is available at a single site (computer/cluster) to a single machine learning algorithm. Novel problems and challenges arise whenever this is not the case and the data is distributed over many sites and must not be collected at a single site, e.g., for privacy reasons. In this section we devise a distributed coRLSR algorithm for this scenario.

Consider a situation in which different companies have similar prediction problems and could greatly benefit from better predictive accuracy. This is, for example, the case for different loan providers each trying to prevent fraud using a certain prediction technique. Another example could be companies

Table 4.1: *Distributed CoRLSR Algorithm*

Input: Matrices as in Optimization Problem 4.1 and Proposition 4.3, or matrices as in Optimization Problem 4.2 and Proposition 4.4.

```
1 Initialize ŷ_u = 0 at each site.
2 repeat
3 for each view v sequentially do
4     α_v ← G_v^{-1} [L_v^T y_v + 2λ U_v^T Σ_{u≠v} ŷ_u]
5     ŷ_v ← U_v α_v
6     send ŷ_v to all sites.
7 until convergence
```

Output: Optimal solution $\boldsymbol{\alpha}_v$ of the respective coRLSR optimization problem.

trying to protect their computers from attacks over the internet using a learned model of internet connections.

In both cases sharing the data or their models could increase the quality of the predictions. However, due to privacy concerns or non-disclosure agreements the companies are rather unlikely to do that. In this section we consider the case that the different companies, however, agree on a set of unlabeled data and to exchange their predictions on this unlabeled data. As the unlabeled data may even be appropriately generated synthetic data, it is realistic to assume that companies do this.

Traditional machine learning algorithms cannot be applied in this setting. We formulate a kernel based optimization problem and show that the globally optimal solution to this problem can be found using gradient descent and only sharing predictions on the unlabeled data.

4.3.1 Block Coordinate Descent CoRLSR

In this section we show that the above non-parametric and semi-parametric coRLSR optimization problems can be solved with an iterative, distributed algorithm that only communicates the predictions of each site about the unlabeled data. The following Propositions 4.3 and 4.4 show that the Optimization Problems 4.1 and 4.2 can also be solved in a distributed setting by Algorithm 4.1. Both propositions are proved subsequently.

Proposition 4.3 *The* non-parametric coRLSR optimization problem *can be solved by Algorithm 4.1 with* $\mathbf{G}_v = \mathbf{L}_v^T\mathbf{L}_v + \eta\mathbf{K}_v + 2(V-1)\lambda\mathbf{U}_v^T\mathbf{U}_v$.

Proposition 4.4 *The* semi-parametric coRLSR optimization problem *can be solved by Algorithm 4.1 with* $\mathbf{G}_v = \mathbf{L}_v^2 + \eta\mathbf{L}_v + 2(V-1)\lambda\mathbf{U}_v^T\mathbf{U}_v$.

With all variables defined as in the corresponding non-parametric and semi-parametric coRLSR definitions and proofs, we can prove both propositions together. Note, however, the slight notational difference between the gradient in the following proof and the gradient in the proof of Proposition 4.1. In the following we use the symmetry of $\mathbf{L}_v$ to replace it by its transpose $\mathbf{L}_v^\mathsf{T}$ for notational harmony with the gradient in the proof of Proposition 4.2.
Proof From the respective proofs we have

$$\nabla_{\boldsymbol{\alpha}_v} Q(\boldsymbol{\alpha}) = 2\mathbf{G}_v\boldsymbol{\alpha}_v - 2\mathbf{L}_v^\mathsf{T}\mathbf{y}_v - 4\lambda \sum_{u:u\neq v} \mathbf{U}_v^\mathsf{T}\mathbf{U}_u\boldsymbol{\alpha}_u.$$

Now, we can compute the gradient directions using predictions ($\hat{\mathbf{y}}_u = \mathbf{U}_u\boldsymbol{\alpha}_u$) on the unlabeled data as

$$\nabla_{\boldsymbol{\alpha}_v} Q_v\left(\boldsymbol{\alpha}_v, \mathbf{y}_v, \{\hat{\mathbf{y}}_u\}_u\right) = 2\mathbf{G}_v\boldsymbol{\alpha}_v - 2\mathbf{L}_v\mathbf{y}_v - 4\lambda\mathbf{U}_v^\mathsf{T} \sum_{u:u\neq v} \hat{\mathbf{y}}_u.$$

While the gradient direction itself is only given jointly

$$-\left(\nabla_{\boldsymbol{\alpha}_1} Q_1\left(\boldsymbol{\alpha}_1, \mathbf{y}_1, \{\hat{\mathbf{y}}_u\}_u\right), \nabla_{\boldsymbol{\alpha}_2} Q_2\left(\boldsymbol{\alpha}_2, \mathbf{y}_2, \{\hat{\mathbf{y}}_u\}_u\right), \ldots\right)^\mathsf{T} \ ,$$

the global minimum can also be found by block coordinate descent (Bertsekas, 1999) over each view v. This only requires setting the block gradient to zero, i.e., solving

$$\mathbf{G}_v\boldsymbol{\alpha}_v = \mathbf{L}_v^\mathsf{T}\mathbf{y}_v + 2\lambda\mathbf{U}_v^\mathsf{T} \sum_{u:u\neq v} \hat{\mathbf{y}}_u \ .$$

As $\mathbf{G}_v$ is strictly positive definite and the objective function is convex, block coordinate descent converges. □

4.3.2 Analysis of Distributed CoRLSR

Block coordinate descent has similar convergence properties as steepest descent (Bertsekas, 1999) which reduces the error rate in each iteration by a factor depending on the largest and the smallest eigenvalue of the Hessian. Assuming that this factor is $1/\delta$, the error after $N \in \mathbb{N}$ iterations is reduced by a factor $1/\delta^N$. Let $n = \max_v n_v$. Given that all labels are from the interval $[-1, 1]$, we can upper bound the starting error $Q(\mathbf{0}) - Q(\boldsymbol{\alpha}^*) \leq Vn$,

where $\boldsymbol{\alpha}^*$ is the optimal solution. Let $\boldsymbol{\alpha}^{(N)}$ be the solution of Algorithm 4.1 after N iterations. To achieve an error reduction factor of at least ϵ, i.e., an upper bound on the error of $Q\left(\boldsymbol{\alpha}^{(N)}\right) - Q(\boldsymbol{\alpha}^*) \leq Vn\epsilon$, we must have $N \geq \log_{1/\delta} \epsilon = \log_\delta \frac{1}{\epsilon}$ iterations.

The matrices $\mathbf{G}_v^{-1}$ in Algorithm 4.1 can be computed in time $O(m^3)$ and $O(mn^2)$ for non-parametric and semi-parametric coRLSR, respectively. It needs to be computed only once and can be computed at the same time for all sites. Step 4 of Algorithm 4.1 can then be computed in time $O\left(V(m+n)^2\right)$ for non-parametric coRLSR and $O(Vmn)$ for semi-parametric coRLSR. As each step has to be performed at each site, each iteration takes $O\left(V^2(m+n)^2\right)$ or $O(V^2mn)$ time. Thus to achieve an error reduction factor of at least ϵ, Algorithm 4.1 takes $O\left(m^3 + V^2(m+n)^2 \left\lceil \log_\Delta \frac{1}{\epsilon} \right\rceil\right)$ and $O\left(mn^2 + V^2mn \left\lceil \log_\delta \frac{1}{\epsilon} \right\rceil\right)$ time, respectively. Similarly, in each iteration, Vm numbers have to be broadcasted. If we consider the machine precision as constant, this requires broadcasting $O\left(Vm \left\lceil \log_\delta \frac{1}{\epsilon} \right\rceil\right)$ bits to achieve an error reduction by the factor ϵ.

4.4 Empirical Evaluation

In this section we summarize experiments comparing regular RLSR with non-parametric and semi-parametric coRLSR on benchmark regression datasets.

In all experiments we use a Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x}-\mathbf{x}'\|^2/\sigma^2)$ with $\sigma^2 = 1/n^2 \sum_{i,j=1}^n \|\mathbf{x}_i - \mathbf{x}_j\|^2$ and $\eta = \left(\sum_{i=1}^n \|\mathbf{x}_i\|/n\right)^{-1}$ as the regularization parameter. Note, that σ and η depend only on the labeled examples; in the case of multiple views, σ_v and η_v are computed from the attributes in the respective view v. We report scaled root mean square errors (rmse)

$$\text{rmse}(f) = \frac{1}{\max y_i} \sqrt{\frac{1}{m} \sum_{i=1}^m (f(\mathbf{x}_i) - y_i)^2}.$$

which allows for viewing all results in the same figure.

4.4.1 UCI Experiments

The UCI repository (Newman et al., 1998) contains 63 data sets with continuous target attributes. We omit data sets containing less than 50 examples and/or less than 4 attributes. We leave out the largest 20 data sets because of memory problems in Matlab with inverting the matrices for the non-parametric case. On the remaining 32 data sets we perform a 10-fold 'inverse' cross validation, i.e., in each run we use one fold as labeled examples

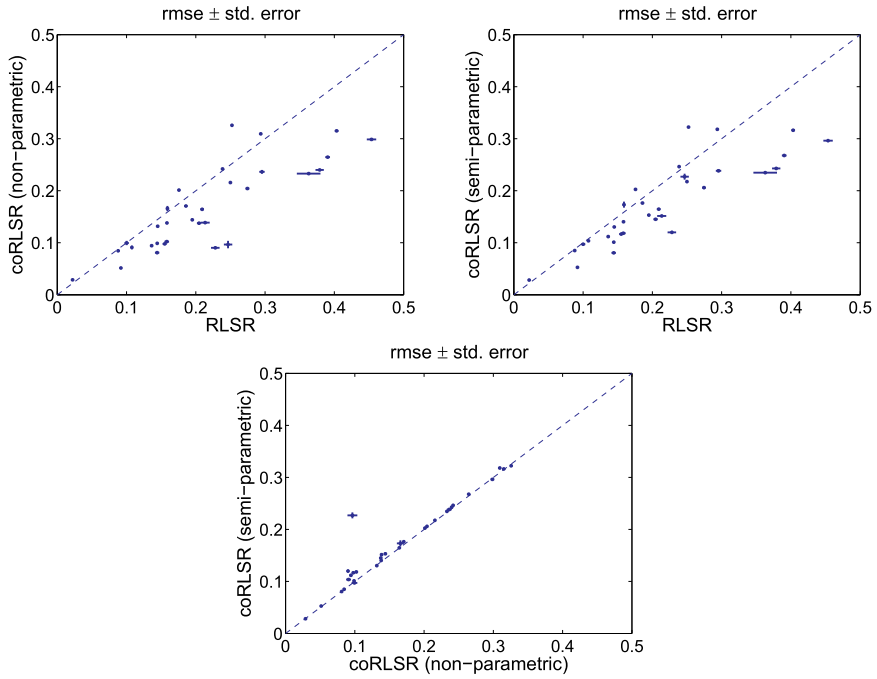


Figure 4.1: *Pairwise* rmse *for non-parametric coRLSR, semi-parametric coRLSR, and regular RLSR over 32 UCI data sets.*

and the other 9 folds as unlabeled and holdout examples. In each run the available attributes are split randomly into two disjoint sets. The results are averages over 20 such runs. In all experiments we use $\lambda = 1/10$. The results are shown in Figure 4.1 where error bars indicate the standard error.

In Figure 4.1 (top left) we plot the rmse of regular RLSR for all 32 UCI problems (x-axis) against the corresponding rmse values of non-parametric coRLSR (y-axis). Thus, each point refers to a UCI problem. The dashed line marks the threshold where both methods perform equally well. Points below this line indicate that non-parametric coRLSR has a lower rmse for these data sets compared to regular RLSR. Figure 4.1 (top right) shows the analog for regular RLSR and semi-parametric coRLSR. Both comparisons show that the multi-view algorithms outperform the baseline in most of the 32 problems. Figure 4.1 (bottom) compares the two multi-view methods. Semi-parametric coRLSR performs slightly worse than non-parametric coRLSR.

While the figures indicate that coRLSR outperforms the baseline RLSR method over all datasets, we want to confirm this hypothesis in a sound statistical test. We use the null hypotheses that the algorithms perform equally well. As suggested recently by Demšar (2006) we use the Wilcoxon signed ranks test.

The Wilcoxon signed ranks test is a nonparametric test used to detect shifts in populations given a number of paired samples. The underlying idea is that under the null hypothesis the distribution of differences between the two populations is symmetric around zero. The procedure is as follows: (*i*) compute the differences between the pairs, (*ii*) determine the ranking of the absolute differences, and (*iii*) sum over all ranks with positive and negative difference to obtain W_+ and W_-, respectively. The null hypothesis can be rejected if W_+ (and W_- depending on whether we need a one-sided or a two-sided test) is located in the tail of the null distribution which has a sufficiently small probability.

The critical value of the one-sided Wilcoxon signed ranks test for 32 samples on a 0.5% significance level is 128. The test statistic for comparing non-parametric coRLSR against RLSR is $54 < 128$, the test statistic for comparing semi-parametric coRLSR against RLSR is $66 < 128$, and finally the test statistic for comparing parametric coRLSR against semi-parametric coRLSR is $63 < 128$. Hence, on this significance level we can reject all three null hypotheses. We conclude that the multi-view algorithms significantly outperform regular RLSR and that non-parametric coRLSR is the best regression method of our study.

4.4.2 Results for KDD Cup 2004 data set

In the KDD Cup data set, the task is to predict the amount of money donated to a charity. The original data set comes with 479 attributes. We use a binary encoding of nominal attributes with fewer than 200 distinct values. Since there are many missing values we add an indicator attribute for each continuous attribute. The indicator equals 1 if the actual value is missing and 0 otherwise. The modified data set contains 95412 training examples with 5551 attributes. We use a resampling approach to adjust λ. For a fixed λ we draw a specified number of labeled and unlabeled examples and distinct holdout examples at random in each iteration. We average the rmse on the holdout set over 25 runs with distinct, randomly drawn attribute splits. We compare an equal number of parameter values for all methods. For each problem we fix the apparently optimal λ for all methods and re-evaluate the rmse for these parameter settings by again averaging over 25 runs with distinct resampled training and holdout sets.

In order to explore the influence of unlabeled examples we use 100 labeled and 200 holdout examples and vary the number of unlabeled examples. The results are shown in Figure 4.2 (left). For 100 labeled and no unlabeled examples both multi-view algorithms have lower rmse compared to the baseline by simply averaging the predictions of the two views. As the number of

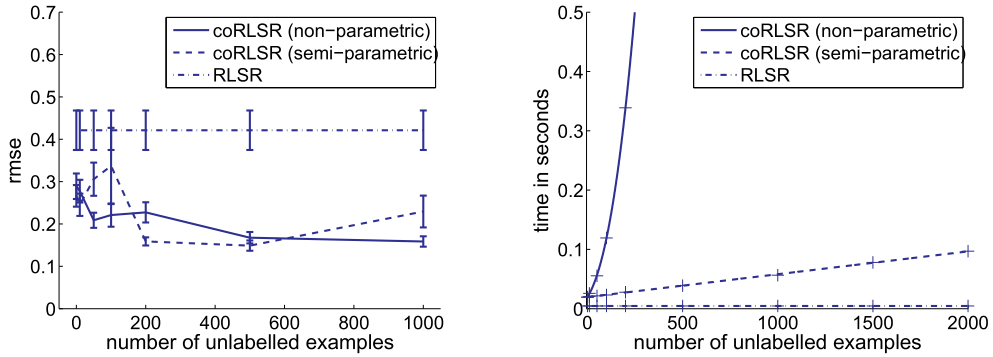


Figure 4.2: *Left: Results on the KDD Cup 98 data set with* 100 *labeled instances and varying numbers of unlabeled ones. Right: Execution times.*

Table 4.2: *Large-scale results on the KDD Cup 98 data set with* 100 *labeled instances and* 10000*,* 50000*, and* 90000 *unlabeled examples.*

	10000	50000	90000
rmse	0.1312 ± 0.006	0.1078 ± 0.004	0.1253 ± 0.006

unlabeled examples increases, the advantage of multi-view over single-view regression increases further. Again, non-parametric coRLSR turns out to be the best regression method.

The performance of semi-parametric coRLSR can be further improved by increasing the number of unlabeled instances. Table 4.2 reports rmse values for 100 labeled and 10000, 50000, and 90000 unlabeled instances. Note that non-parametric coRLSR is not feasible for these sample sizes.

Figure 4.2 (right) compares the execution time of regular RLSR, non-parametric, and semi-parametric coRLSR. The figure shows execution time for a fixed number of labeled and different numbers of unlabeled examples and fitted polynomials. The empirical results confirm our theoretical findings. Non-parametric coRLSR is costly in terms of computation time (the degree of the fitted polynomial is 3). Its approximation is considerably faster (the fit of semi-parametric coRLSR is a linear function of the number of unlabeled examples as shown in Proposition 4.2). For any number of unlabeled datapoints, the runtime of semi-parametric coRLSR is comparable to that of regular RLSR.

4.5 Conclusions

In this chapter we proposed co-regularized least squares regression (coRLSR), a novel semi-supervised regression algorithm based on the co-learning framework. While coRLSR like many other semi-supervised or transductive approaches has cubic runtime complexity in the amount of unlabeled data, we proposed a semi-parametric approximation of coRLSR which scales linearly with the amount of unlabeled data.

Both non-parametric and semi-parametric coRLSR have closed form solutions in the usual centralized learning setting. Additionally, both can be solved in the less common distributed learning setting where the labeled data must not be joined at a single site. This can be achieved by an iterative distributed algorithm that only communicates the predictions about the unlabeled data at each iteration.

Empirical results showed that coRLSR as well as its semi-parametric approximation clearly outperform traditional regularized least squares regression even on problems where there is no given feature split. The observed improvements were achieved by applying co-learning based on a random feature split which therefore might even be more pronounced when natural groups of features are available. While non-parametric coRLSR outperforms its semi-parametric approximation in predictive accuracy, semi-parametric coRLSR is clearly more desireable than the exact, non-parametric, version in terms of execution time.

Chapter 5

Co-perceptrons

In this chapter we leave the field of univariate predictions models and turn towards predicting structured output variables with a generalized variant of the perceptron algorithm (Collins and Duffy, 2002; Altun et al., 2003b). The structural perceptron algorithm minimizes the empirical risk for 0/1 loss and allows a dual representation that depends only on inner products in joint input-output spaces. The perceptron allows a dual representation where the inner product in joint input-output space can efficiently be computed by kernel functions.

We study a semi-supervised variant of the structured perceptron. The co-perceptron inherits the advantages of its fully supervised counterpart, that is, it also minimizes the 0/1 loss and allows for the use of kernel functions. Moreover, unlabeled examples can be included into the training process by taking a multi-view approach. The semi-supervised perceptron implements the consensus maximization principle. Although the (semi-supervised) perceptron relies on a simple but intuitive algorithm, the key insights of this chapter form the basis of generalized semi-supervised support vector learning in Chapter 6. Readers who are not familiar with perceptron learning may find a review of the binary perceptron in Appendix A helpful.

Section 5.1 reports on related work. For the reader's convenience we briefly introduce the generalized variant of the single-view perceptron (Collins and Duffy, 2002; Altun et al., 2003b) in Section 5.2 before presenting its semi-supervised generalization in Section 5.3. Empirical studies with named entity recognition problems are presented in Section 5.4. Section 5.5 concludes.

5.1 Related Work

In a rapidly developing line of research, many variants of discriminative sequence models are currently being explored. Recently studied variants include maximum entropy Markov models (McCallum et al., 2000), conditional random fields (Lafferty et al., 2001), perceptron re-ranking (Collins, 2002), hidden Markov support vector machines (Altun et al., 2003b), label sequence boosting (Altun et al., 2003a), max-margin Markov models (Taskar et al., 2004a), case-factor diagrams (McAllester et al., 2004), sequential Gaussian process models (Altun et al., 2004a), kernel conditional random fields (Lafferty et al., 2004), and support vector machines for structured output spaces (Tsochantaridis et al., 2004).

De Sa (1994) observes a relationship between consensus of multiple hypotheses and their error rates and devises a semi-supervised learning method by cascading multi-view vector quantization and linear classification. A multi-view approach to word sense disambiguation combines a classifier that refers to the local context of a word with a second classifier that utilizes the document in which words co-occur (Yarowsky, 1995). Blum and Mitchell (1998) introduce the co-training algorithm for semi-supervised learning that greedily augments the training set of two classifiers. A version of the AdaBoost algorithm boosts the agreement between two views on unlabeled data (Collins and Singer, 1999).

Dasgupta et al. (2001) and Abney (2002) give PAC bounds on the error of co-training in terms of the disagreement rate of hypotheses on unlabeled data in two independent views. This justifies the direct minimization of the disagreement. The co-EM algorithm for semi-supervised learning probabilistically labels all unlabeled examples and iteratively exchanges those labels between two views (Nigam and Ghani, 2000; Ghani, 2002). Muslea et al. (2002) extend co-EM for active learning and Brefeld and Scheffer (2004) examine a co-EM wrapper for the support vector machine.

The generalized perceptron is highly related to ranking tasks (Freund et al., 1998; Collins, 2000) since the goal is to assign the highest value to the relevant (true) output. The single-view perceptron for structured output spaces was introduced by Collins and Duffy (2002). They used convolution kernel functions (Haussler, 1999) for natural language parsing. Earlier models for parsing relied on hand crafted grammars (Johnson et al., 1999), re-ranking (Collins, 2000), or maximum likelihood estimates (Johnson, 1998). Altun et al. (2003b) leverage this approach to support vector machines and explore label sequence learning tasks with implicit 0/1 loss.

5.2 Generalized Perceptrons

The perceptron for structured output spaces (Collins and Duffy, 2002; Altun et al., 2003b) is a generalized linear model

$$f(x, y) = \langle \mathbf{w}, \Phi(x, y) \rangle, \tag{5.1}$$

where $\Phi(x, y)$ denotes the joint feature representation of input x and output y. Given the i-th training example (x_i, y_i), an update is performed if an alternative output $\bar{y}$ is decoded instead of the correct y_i, that is

$$y_i \neq \hat{y} = \operatorname*{argmax}_{\bar{y} \in \mathcal{Y}(x)} \langle \mathbf{w}, \Phi(x, \bar{y}) \rangle.$$

This is equivalent to requiring that the inner product of the weights and the differences of the feature representations of the true pair $\Phi(x_i, y_i)$ and all alternative labelings $\bar{y}$ to be positive, that is,

$$\max_{\substack{\bar{y} \in \mathcal{Y}(x_i) \\ \bar{y} \neq y_i}} \langle \mathbf{w}, \Phi(x_i, y_i) \rangle - \langle \mathbf{w}, \Phi(x_i, \bar{y}) \rangle = \max_{\substack{\bar{y} \in \mathcal{Y}(x_i) \\ \bar{y} \neq y_i}} \langle \mathbf{w}, \Phi(x_i, y_i) - \Phi(x_i, \bar{y}) \rangle > 0$$

must hold for all training instances $1 \leq i \leq n$. The generalized update rule (see also Equation A.10) inherits these difference vectors and has the form

$$\mathbf{w} \leftarrow \mathbf{w} + \Phi(x_i, y_i) - \Phi(x_i, \bar{y}). \tag{5.2}$$

The influence of the correct feature representation is increased while that of the alternative output is decreased. The update rule 5.2 together with the initialization $\mathbf{w} = \mathbf{0}$ allows us to rewrite the weight vector in terms of differences between the training pairs and erroneously decoded outputs

$$\mathbf{w} = \sum_i \sum_{\substack{\bar{y} \in \mathcal{Y}(x_i) \\ \bar{y} \neq y_i}} \alpha_{i y_i \bar{y}} \left(\Phi(x_i, y_i) - \Phi(x_i, \bar{y}) \right), \tag{5.3}$$

where the $\alpha_{i y_i \bar{y}} \in \mathbb{N}_0$ act as a counter, detailing how many times the alternative output $\bar{y}$ has been decoded instead of the correct output y_i of the i-th example. Here, the true output as second substript is redundant since it is involved in every update associated with the i-th example but its benefit will become obvious in the next section. We keep it for the sake of consistency instead of changing the notation at a later time.

Altun et al. (2003b) use a slightly different notation and propose the use of pseudo-labels $z_i(\bar{y})$ that equal $+1$ if $\bar{y} = y_i$ and -1 otherwise. In terms of pseudo-labels, Equation 5.3 can be written as

$$\mathbf{w} = \sum_i \sum_{\bar{y} \in \mathcal{Y}(x_i)} \alpha_i'(\bar{y}) z_i(\bar{y}) \Phi(x_i, \bar{y}),$$

where $\alpha' \in \mathbb{N}_0$ are adjusted counts. Notice that the notations are equivalent and can be translated into each other using the equations $\alpha'_i(y_i) = \sum_{\bar{y} \neq y_i} \alpha_{iy_i\bar{y}}$ and $\alpha'(\bar{y}) = \alpha_{iy\bar{y}}$ for all $\bar{y} \neq y_i$. Substituting Equation 5.3 into Equation 5.1 leads to its equivalent dual formulation

$$f(x, y) = \sum_i \sum_{\substack{\bar{y} \in \mathcal{Y}(x_i) \\ \bar{y} \neq y_i}} \alpha_{iy_i\bar{y}} \Big(\langle \Phi(x_i, y_i), \Phi(x, y) \rangle - \langle \Phi(x_i, \bar{y}), \Phi(x, y) \rangle \Big). \tag{5.4}$$

The dual depends only on inner products in input-output space and can be computed efficiently by means of dual variables $\alpha_{iy_i\bar{y}} \in \mathbb{N}_0$ and a composite kernel $k'(x_i, y_i, \bar{y}, x, y) = k(x_i, y_i, x, y) - k(x_i, \bar{y}, x, y)$, with $k(x, y, x', y') = \langle \Phi(x, y), \Phi(x', y') \rangle$. Thus, we can rewrite Equation 5.4 as

$$f(x, y) = \sum_i \sum_{\substack{\bar{y} \in \mathcal{Y}(x_i) \\ \bar{y} \neq y_i}} \alpha_{iy_i\bar{y}} \, k'(x_i, y_i, \bar{y}, x, y). \tag{5.5}$$

Algorithm 5.1 shows the dual perceptron algorithm that consecutively decodes each input in the training sample. When the decoding yields an incorrectly labeled sequence $\bar{y}$ for the i-th example, instead of the correct sequence y_i, then the corresponding $\alpha_{iy\bar{y}}$ is updated according to

$$\alpha_{iy_i\hat{y}} \leftarrow \alpha_{iy_i\hat{y}} + 1. \tag{5.6}$$

After an error has occurred, the correct sequence receives more, and the incorrect prediction receives less influence. Since all initial $\alpha_{iy_i\bar{y}} = 0$, it suffices to store only those sequences in memory that have been used for an update.

A simple generalization of Novikoff's theorem (Novikoff, 1962), gives convergence guarantees of the perceptron in terms of an upper bound on the number of update steps given by $t \leq (r/\bar{\gamma})^2 \|\mathbf{w}\|^2$, where r denotes the the radius of the smallest hypersphere enclosing all difference vectors (Collins, 2002; Altun et al., 2003b). As in the regular case, for the theorem to hold a solution preserving a functional margin $\bar{\gamma} > 0$ must exist.

5.3 Co-perceptrons

In this section we extend the perceptron for structured output variables to semi-supervised learning. We focus on two-view learning, however, the presented results are easily generalized to V-view learning with $V \geq 1$. The co-perceptron implements the consensus maximization principle, that is, it

Table 5.1: *Dual Perceptron Algorithm*

Input: Labeled data $(\mathcal{x}_1, y_1), \ldots, (\mathcal{x}_n, y_n)$.

```
Initialize all α_{i y_i ȳ} = 0.
repeat
for i = 1, ..., n
    ŷ = argmax_{ȳ ∈ Y(x_i)} f(x_i, ȳ) = argmax_ȳ Σ_j Σ_{ȳ_j ≠ y_j} α_{j y_j ȳ} k(x_j, y_j, ȳ_j, x_i, ȳ)
    if y_i ≠ ŷ then
        Increment α_{i y_i ŷ} ← α_{i y_i ŷ} + 1
    end
end
until convergence
```

Output: Trained hypothesis $f(\mathcal{x}, y)$.

minimizes the error on the labeled examples and maximizes the agreement between two hypotheses on the unlabeled examples.

In the semi-supervised setting we are given a set of n labeled examples $(\mathcal{x}_1, y_1), \ldots, (\mathcal{x}_n, y_n)$ and m unlabeled examples $\mathcal{x}_{n+1}, \ldots, \mathcal{x}_{n+m}$. We model the views by utilizing two joint feature representations $\Phi^1(\mathcal{x}_i, y_i)$ and $\Phi^2(\mathcal{x}_i, y_i)$ that live in distinct vector spaces. We assume that both feature maps are sufficient for the decoding strategy. The joint decision function is defined as the sum of the decision functions in the two views,

$$f(\mathcal{x}, y) = f^1(\mathcal{x}, y) + f^2(\mathcal{x}, y). \tag{5.7}$$

In each view relation 5.3 holds and either decision function has the form

$$f^v(\mathcal{x}, y) = \sum_{i=1}^{n+m} \sum_{\substack{\bar{y} \in \mathcal{Y}(\mathcal{x}_i) \\ \bar{y} \neq y_i}} \alpha^v_{i y_i \bar{y}} \, k^v(\mathcal{x}_i, y_i, \bar{y}, \mathcal{x}, y), \tag{5.8}$$

where $v = 1, 2$. The composite kernel k^v is defined analogously to the single-view case for $v = 0, 1$ by

$$k^v(\mathcal{x}_i, y_i, \bar{y}, \mathcal{x}', y') = \langle \Phi^v(\mathcal{x}_i, y_i), \Phi^v(\mathcal{x}', y') \rangle - \langle \Phi^v(\mathcal{x}_i, \bar{y}), \Phi^v(\mathcal{x}', y') \rangle. \tag{5.9}$$

According to the consensus maximization principle, the semi-supervised perceptron algorithm now has to minimize the number of errors for labeled examples and the disagreement for unlabeled examples.

Table 5.2: *Dual Co-perceptron Algorithm*

Input: Labeled $(x_1, y_1), \ldots, (x_n, y_n)$ and unlabeled data $x_{n+1}, \ldots, x_{n+m}$, parameter C_u, number of iterations t_{max}.

```
Initialize all α^v_{i y_i ȳ} = 0, v = 1,2.
for t = 1,...,t_max
    for i = 1,...,n+m
        Retrieve ŷ^1 = argmax_{y∈Y(x_i)} Σ_{j=1}^{n+m} Σ_{ȳ≠y_j} α^1_{j y_j ȳ} k^1(x_j, y_j, ȳ, x_i, y).
        Retrieve ŷ^2 = argmax_{y∈Y(x_i)} Σ_{j=1}^{n+m} Σ_{ȳ≠y_j} α^2_{j y_j ȳ} k^2(x_j, y_j, ȳ, x_i, y).
        if i–th example is a labeled example
            if y_i ≠ ŷ^v, v = 1,2
                Update: α^v_{i y_i ŷ^v} ← α^v_{i y_i ŷ^v} + 1
            end
        else
            if ŷ^1 ≠ ŷ^2
                Update: α^1_{j ŷ^2 ŷ^1} ← α^1_{j ŷ^2 ŷ^1} + C_u and α^2_{j ŷ^1 ŷ^2} ← α^2_{j ŷ^1 ŷ^2} + C_u}
            end
        end
    end
end
```

Output: Combined hypothesis $f(x, y) = f^1(x, y) + f^2(x, y)$.

Each view $v = 0, 1$ predicts the output $\hat{y}^v$ for an input i, whether it is labeled or unlabeled, analogously to the single-view perceptron according to

$$\hat{y}^v = \underset{\bar{y} \in \mathcal{Y}(x_i)}{\operatorname{argmax}} f^v(x_i, \bar{y}). \tag{5.10}$$

For labeled examples the goal is to minimize the error of either hypothesis. Thus, the perceptron update rule for labeled examples remains unchanged; if view v misclassifies the i-th labeled example, that is $y_i \neq \hat{y}^v_j$, then the respective parameters are updated according to Equation 5.11,

$$\alpha^v_{i y_i \hat{y}^v_j} \leftarrow \alpha^v_{i y_i \hat{y}^v_j} + 1. \tag{5.11}$$

If the views disagree on the j-th unlabeled example – that is, $\hat{y}^1_j \neq \hat{y}^2_j$ – updates have to be performed that reduce the discord. Intuitively, each decision is passed towards that of the peer view. In other words, we treat

the prediction of each view v as true label for the peer view $\bar{v}$. Therefore, in the case of a disagreement on the j-th unlabeled example both views are updated simultaneously according to Equation 5.12.

$$\alpha^1_{j\hat{y}^2_j\hat{y}^1_j} \leftarrow \alpha^1_{j\hat{y}^2_j\hat{y}^1_j} + C_u; \quad \alpha^2_{j\hat{y}^1_j\hat{y}^2_j} \leftarrow \alpha^2_{j\hat{y}^1_j\hat{y}^2_j} + C_u \tag{5.12}$$

The parameter $0 \leq C_u \leq 1$ determines the influence of a single unlabeled example. If $C_u = 1$, each example has the same influence whether it is labeled or unlabeled. Once the co-perceptron is trained, the joint decision function (Equation 5.7) predicts outputs for unseen, new inputs $\mathcal{x}$. Algorithm 5.2 shows the co-perceptron algorithm.

5.4 Empirical Results

In our empirical evaluation we concentrate on named entity recognition (NER) problems. We study the benefit of semi-supervised label sequence learning and compare the co-perceptron to its single-view counterpart and a hidden Markov model. The latter is trained with multi-Bernoulli distributed observation probabilities. We use the data set provided for task 1A of the Biocreative challenge (Yeh et al., 2005) and the Spanish news wire article corpus of the shared task of CoNLL 2002 (Sang, 2002).

In each experiment we randomly draw a specified number of labeled and unlabeled training and holdout sentences without replacement in each iteration. We assure that each label occurs at least once in the labeled training data; otherwise, we discard and draw again. We first optimize parameter C_u using resampling; we then fix C_u and present curves that show the average token-based error over 100 randomly drawn training and holdout sets. The baseline methods HMM and single-view perceptron are trained on concatenated views; error bars indicate standard error.

5.4.1 Biocreative Data Set

The Biocreative data contains 7500 sentences from biomedical papers randomly selected from the PubMed/MedLine data base. The goal is to recognize gene and protein names. We discriminate tokens that are parts of gene names against all other tokens that are annotated with part-of-speech tags according to the Penn tree bank (Marcus et al., 1993). View 1 consists of the token itself together with letter 2, 3, and 4-grams; view 2 contains surface clues like capitalization, inclusion of Greek symbols, numbers, and others as documented in Table 5.3. Hakenberg et al. (2005) detail the usefullness of

Table 5.3: *Feature Classes used in the Biocreative Experiments*

Feature	Example
Token	Sro7
$2, 3, 4$-grams of token	
initial uppercase	Msp
all chars are upper case	MMTV
mixed upper and lower case	kDa
digit, dot, digit	75.0
numbers	HSF1
special symbols	ICAM-1
greek symbols	alpha
length of token	
positions of upper case characters	

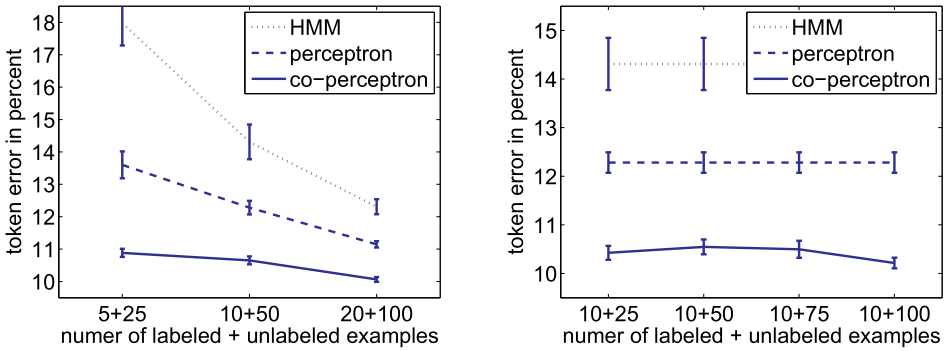


Figure 5.1: *Perceptron learning curves for Biocreative. Displayed are HMM (dotted), single-view perceptron (dashed), and co-perceptron (solid).*

the extracted features by a recursive feature elimination approach. In both views we draw features from a window of length 5.

Figure 5.1 (left) shows learning curves for HMM, single-view and multi-view perceptrons for the Biocreative task. The two perceptrons significantly outperform the generative HMM. Except for one point, the co-perceptron beats its single-view, purely supervised counterpart significantly. In Figure 5.1 (right) we vary the number of unlabeled sequences for the Biocreative data set. As the number of unlabeled data increases, the advantage of multi-view over single-view sequence learning increases further. The co-perceptron utilizes the additional unlabeled examples effectively.

Table 5.4: *Feature classes used in the Spanish news wire experiments*

Feature	Example
Token	informaciones
initial uppercase	Madrid
initial lowercase	especial
uppercase at position t	CrimeNet
lowercase at position t	especial
digit, dot, digit	34.32
numbers	1975
special symbols	Lasa-Zabala
token between brackets/quotes	el Grupo ¨Magenta¨
initial token	Por ...

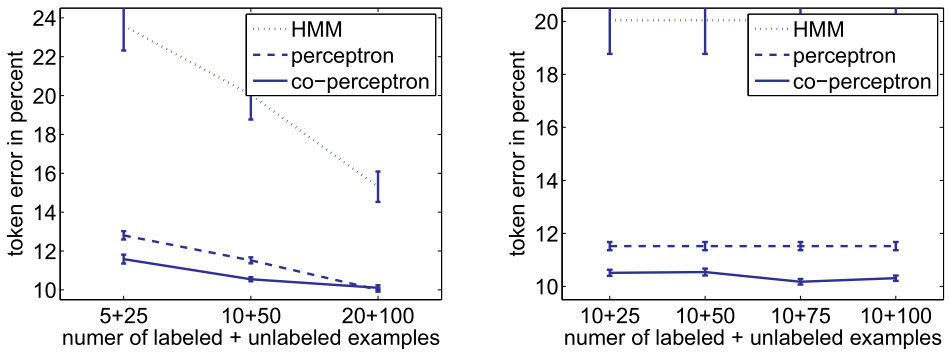


Figure 5.2: *Perceptron learning curves for Spanish news wire. Displayed are HMM (dotted), single-view perceptron (dashed), and co-perceptron (solid).*

5.4.2 Spanish News Wire

The CoNLL2002 data consists of sentences from a Spanish news wire archive and contains 9 label types which distinguish person, organization, location, and other names. We utilize all 9 labels for our experiments. We use 3100 sentences of between 10 and 40 tokens; leading to approximately 24000 distinct tokens in the dictionary. Moreover, we extract surface clue features such as capitalization, numbers, and special symbol features (see Table 5.4 for examples). We represent each sentence by a token view and a view of surface clues.

Figure 5.2 (left) details the learning curves for the Spanish news wire data set. Again, the HMM performs most poorly and the co-perceptron

outperforms the single-view perceptron significantly. In Figure 5.2 (right) we fix the number of labeled examples and vary the number of unlabeled examples. We observe a similar effect as for Biocreative data set. Increasing the number of unlabeled instances leads to lower error rates. Notice that for 5 labeled and 25 unlabeled examples the difference between both perceptrons and the hidden Markov model is larger than 10%.

5.4.3 Execution Time

Figure 5.3 (left) plots execution time against training set size. The performance benefits are at the cost of significantly longer training processes. The observed execution time of co-perceptron scales between linearly and quadratically with the number of unlabeled examples.

5.4.4 Feature splits

So far, we split the available features into a token and letter n-gram view and a surface clue view. In this section, we compare this split to a random feature split and to a split in which view 1 contains all odd, and view 2 all even features. Hence, each view contains half of the token features as well as half of the surface clues. We utilize the Spanish news wire data set and average the results from over 100 runs with randomly drawn labeled and unlabeled instances for each split and method. We also draw distinct random feature splits in each repetition.

Surprisingly, Figure 5.3 (right) shows that random splits work slightly better than the token versus surface clues split. Moreover, Figure 5.3 (right) shows that the odd-even split performs significantly better in two out of three cases than the token versus surface clue split.

Hence, our experiments show that even though multi-view learning using the split of token versus surface clues leads to a substantial improvement over single-view learning, a random or odd-even split lead to an even better performance. Since the performance of our initial split is below average, it is clear that there has to be a feature split that lies above the average. Finding such a split remains an open challenge.

5.5 Conclusions

Starting from the structural perceptron, we constructed a semi-supervised learning method in joint input-output spaces. We derive the co-perceptron

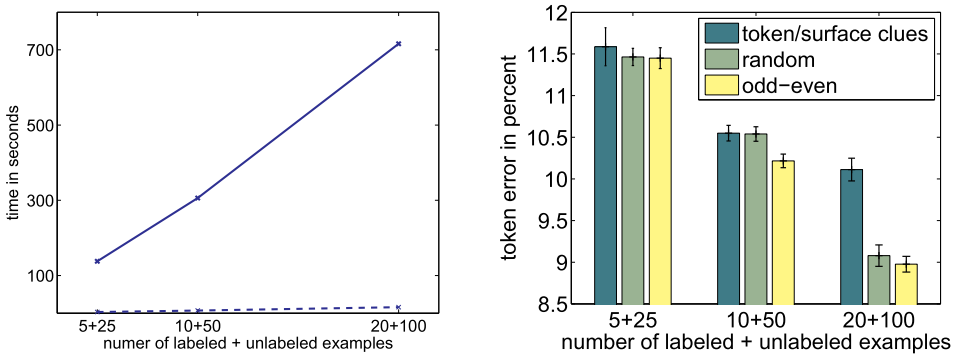


Figure 5.3: *Left: Execution time for single-view perceptron (dashed) and co-perceptron (solid). Right: Error for different splits of features into views for Spanish news wire.*

that implements the principle of consensus maximization between hypotheses. Our experiments with named entity recognition problems show that, on average, this method utilizes unlabeled data effectively and significantly outperforms their purely supervised counterparts structured perceptron and HMM.

Although we focus on two-view learning, all results in this chapter can easily be generalized to the V-view case with $V \geq 1$. In this case, the rate of disagreement is evaluated pairwise between all views as proposed in Chapter 4. We derive a self-training variant of the single-view perceptron in the case $v = 1$ and the supervised perceptron in the case $v = 1$ and $m = 0$. Moreover, co-perceptron generalizes the binary perceptron.

We observed that random feature splits perform better than splitting the features into a token view and a view of surface clues. Nevertheless, the multi-view perceptron outperforms its supervised counterparts even for the initial weak split. Finding a good feature split into two (or more) views is hardly tractable since one has to take all possible partitionings or clusterings into account. From a statistical point of view, this is equivalent to finding a factorization of the attributes that fulfills some criteria, for instance a high within-cluster correlation and a small between-cluster correlation (Jakulin and Bratko, 2004).

Chapter 6

Co-support Vector Learning

In this chapter we present semi-supervised support vector machines for structured output spaces (coSVMs). Similar to the co-perceptron, coSVMs implement the consensus maximization principle but take a large margin approach in joint input-output space. The latter leads to confident predictions and a solid theoretical underpinning. Moreover, coSVMs allows the incorporation of arbitrary loss functions and provide sparse solutions in the number of used constraints.

We begin with a review of related work in Section 6.1 and the structural SVM (Tsochantaridis et al., 2005) in Section 6.2 that extends the binary support vector machine (Appendix A) to structured output variables. The semi-supervised coSVM is then presented in Section 6.3 in greater detail for slack-rescaling losses. We provide details on the optimization strategy in Section 6.4 and report the empirical results in Section 6.5. Section 6.6 concludes.

6.1 Related Work

Due to the success of discriminative structured prediction models, several algorithms have been explored that utilize joint spaces of input and output variables; these include max-margin Markov models (Taskar et al., 2004a), kernel conditional random fields (Lafferty et al., 2004), Gaussian processes for segmenting and annotating sequences (Altun et al., 2004a), and support vector machines for structured output spaces (Tsochantaridis et al., 2005). These methods utilize kernels to compute the inner product in input-output space. Not only does this approach allow us to capture long-distance dependencies between inputs and outputs, but it is also sufficiently versatile to cover other structures of outputs, such as (parse) trees, lattices, or

graphs. An application-specific learning method is constructed by defining appropriate features, and choosing a decoding procedure that efficiently calculates the argmax, exploiting the dependency structure of the features. For instance, the decoder can be a chart-parser for tree-structured outputs, or a Viterbi algorithm when joint features are constrained to depend only on adjacent outputs. Nguyen and Guo (2007) compare state-of-the-art approaches in sequential learning and Rätsch and Sonnenburg (2007) propose efficient decomposition techniques allowing for large-scale experiments.

Multi-view methods naturally allow for the inclusion of unlabeled examples in discriminative learning. The co-training (Blum and Mitchell, 1998) and co-EM algorithms (Nigam and Ghani, 2000) iteratively increment the consensus of independent hypotheses by exchanging conjectured labels for unlabeled data. Recently, Farquhar et al. (2006) proposed a fully supervised variant of a co-support vector machine that minimizes the training error as well as the disagreement between two views. Dasgupta et al. (2001) give PAC bounds on the error of co-training in terms of the disagreement rate of hypotheses on unlabeled data in two independent views. A corollary of their results that holds under general assumptions is the inequality

$$Pr(f^1 \neq f^2) \geq \max\{Pr(err(f^1)), Pr(err(f^2))\}.$$

That is, the probability that two independent hypotheses disagree upper bounds the error rate of either hypothesis. Thus, the strategy of semi-supervised multi-view learning can be stated as: Minimize the error for labeled examples and maximize the agreement for unlabeled examples.

Altun et al. (2006) propose a semi-supervised approach to label sequence learning by integrating Laplacian priors into structured large margin classifiers for pitch accent prediction. Xu et al. (2006) derive unsupervised M^{3}Networks by employing SDP relaxation techniques and Lee et al. (2007) study semi-supervised CRFs and include unlabeled data via an entropy criterion.

6.2 SVMs for Structured Output Variables

Before we present the derivation of the SVM for structured output variables recall the generalized perceptron in Chapter 5 whose goal is to learn a linear discriminant function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ given by $f(x, y) = \langle \mathbf{w}, \Phi(x, y) \rangle$ that correctly decodes any output y_i of the training sample (x_i, y_i); that is,

$$y_i = \operatorname*{argmax}_{\bar{y} \in \mathcal{Y}} f(x_i, \bar{y}).$$

This is the case if Equation 6.1 holds for all $1 \leq i \leq n$.

$$f(x_i, y_i) - \max_{\substack{\bar{y} \in \mathcal{Y}(x_i) \\ \bar{y} \neq y_i}} f(x_i, \bar{y}) > 0 \tag{6.1}$$

We make Equation 6.1 slightly more conservative by incorporating a confidence tube around the decision hyperplane or, in other words, we take a large margin approach and require the existence of $\bar{\gamma} \geq 0$ such that

$$\forall_{i=1}^{n} f(x_i, y_i) - \max_{\substack{\bar{y} \in \mathcal{Y}(x_i) \\ \bar{y} \neq y_i}} f(x_i, \bar{y}) \geq \bar{\gamma} \tag{6.2}$$

holds. The scalar $\bar{\gamma}$ is called the *functional margin* and depends on the scaling of $\mathbf{w}$. Its analogue, the *geometrical margin*, is independent of the weights and given by $\gamma = \frac{\bar{\gamma}}{\|\mathbf{w}\|}$. Support vector machines enforce confident predictions by maximizing the geometrical margin γ; setting $\bar{\gamma} = 1$ leads us directly to the following hard margin optimization problem.

Optimization Problem 6.1 (Primal Hard Margin SVM) *Given n labeled examples $(x_1, y_1), \ldots, (x_n, y_n)$; the primal hard margin support vector optimization problem is defined as*

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

subject to the constraints $\forall_{i=1}^{n}$, $\forall_{\bar{y} \neq y_i} \langle \mathbf{w}, \Phi(x_i, y_i) - \Phi(x_i, \bar{y}) \rangle \geq 1$.

The shortcoming of Optimization Problem 6.1 is its divergence in case that at least one of the constraints cannot be fulfilled. Thus, in general, we have to allow pointwise relaxations of the hard margin constraints by slack variables. There are several ways of introducing slack variables. For instance, slack variables may penalize every margin violation for every constraint which is equivalent to introducing a slack $\xi(\bar{y})$ for every $\bar{y} \neq y_i$ (Weston and Watkins, 1998; Har-Peled et al., 2002). Another possibility is to bind each slack variable to an input example (Crammer and Singer, 2001) and let ξ_i take the maximal violation caused by that example. We follow the latter approach since it leads to a tight upper bound on the empirical risk. Each slack variable ξ_i is bound to an input example x_i, leading to the following soft-margin optimization problem.

Optimization Problem 6.2 (Primal Soft Margin SVM) *Given labeled examples, joint feature mapping Φ, let $C > 0$ and $r = 1, 2$; the primal soft-margin support vector optimization problem is defined as*

$$\min_{\mathbf{w}, \boldsymbol{\xi}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{r}\sum_{i=1}^{n} \xi_i^r$$

subject to the constraints

$$\forall_{i=1}^{n}, \forall_{\bar{y}\neq y_i} \ \langle \mathbf{w}, \Phi(x_i, y_i) - \Phi(x_i, \bar{y})\rangle \geq 1 - \xi_i$$
$$\forall_{i=1}^{n} \ \xi_i \geq 0.$$

The parameter $r = 1, 2$ denotes a linear or quadratic penalization of the error, respectively, and $C > 0$ determines the trade-off between margin maximization and error minimization. The objective is precisely the regularized empirical risk with hinge loss. The sum $\sum_i \xi_i$ upper bounds the empirical risk with common 0/1 loss. This, however, might not be the best choice for several applications (Joachims, 2005). Recently, two distinct ways of integrating a loss function $\Delta : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}_0^+$ into structured optimization problems have been discussed, a margin-rescaling approach (Taskar et al., 2004a) and a slack-rescaling approach (Tsochantaridis et al., 2005).

Margin-rescaling can be intuitively motivated by recalling that the size of the margin $\gamma = \bar{\gamma}/\|\mathbf{w}\|$ quantifies the confidence in rejecting an erroneously decoded output $\bar{y}$. Reweighting $\bar{\gamma}$ with the current loss $\Delta(y_i, \bar{y}_i)$ leads to a weaker rejection confidence when y_i and $\bar{y}$ are similar, while severe differences among them imply a large rejection threshold. Thus, rescaling the margin by the loss implements the intuition that the confidence of rejecting a mistaken output is proportional to its error. As it will turn out later, margin-rescaling can always be applied when the loss function decomposes over the latent variables of the output structure. This assumption on the loss function leads to a simple integration of the loss into the decoding of the best runner-up. The SVM with margin-rescaling is detailed in Optimization Problem 6.3 and equivalent to applying the augmented hinge loss ℓ_Δ^m in Equation 2.5.

Optimization Problem 6.3 (SVM, Margin-rescaled Loss) *Given n labeled examples, decomposable loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_0^+$, joint feature mapping Φ, tradeoff $C > 0$, and $r = 1, 2$; the primal SVM optimization problem with margin-rescaling loss is defined as*

$$\min_{\mathbf{w},\boldsymbol{\xi}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{r}\sum_{i=1}^{n} \xi_i^r$$

subject to the constraints

$$\forall_{i=1}^{n}, \forall_{\bar{y}\neq y_i} \ \langle \mathbf{w}, \Phi(x_i, y_i) - \Phi(x_i, \bar{y})\rangle \geq \sqrt[r]{\Delta(y_i, \bar{y})} - \xi_i$$
$$\forall_{i=1}^{n} \ \xi_i \geq 0.$$

Table 6.1: *SVM Optimization Algorithm with Slack-rescaling*

Input: i-th labeled example (x_i, y_i), $S^v_{j\neq i}$, C, r, repetitions r_{max}.

repeat
  Initialize $S^v_i = \emptyset$, $\alpha^v_{iy_i\bar{y}} = 0$ **for** all $\bar{y} \in \mathcal{Y}(x_i)$
  $\bar{y}^v_i = \text{argmax}_{\bar{y}\neq y_i}(1 - \langle \mathbf{w}^v, \Phi^v_{i,y_i,\bar{y}}\rangle)\sqrt[r]{\Delta(y_i,\bar{y})}$
  $\xi^v_i = \max_{\bar{y}\in S^v_i}\{(1 - \langle \mathbf{w}, \Phi^v_{i,y_i,\bar{y}}\rangle)\sqrt[r]{\Delta(y_i,\bar{y})}\}$
  if $\langle \mathbf{w}^v, \Phi^v_{i,y_i,\bar{y}^v_i}\rangle < 1 - \frac{\xi^v_i}{\sqrt[r]{\Delta(y_i,\bar{y}^v_i)}}$
    $S^v_i = S^v_i \cup \{\bar{y}^v_i\}$
    Optimize $\alpha^v_{iy_i\bar{y}}$ over all $\bar{y} \in S^v_i$ **with** $S^v_{j\neq i}$ fixed
    $\forall \bar{y} \in S^v$ **with** $\alpha^v_{iy_i\bar{y}} = 0$: $S^v_i = S^v_i \backslash \{\bar{y}\}$
  end
until margin constraint satisfied **or** r_{max} repetitions

Output: Optimized α^v_i and working set S^v_i.

Margin-rescaling may lead to a situation where the contribution of an example to the global solution depends only on irrelevant outputs $\bar{y}$ since the maximum over erroneous outputs $\max_{\bar{y}\neq y_i}(\langle \mathbf{w}, \Phi(x_i, \bar{y})\rangle + \Delta(y_i, \bar{y}))$ may be dominated by the loss function. In this case only those examples are decoded that realize a high loss, irrespectively of their appropriateness determined by the inner product.

A different approach is taken by Tsochantaridis et al. (2005), who propose to rescale the slack variables instead of the functional margin; thus, the error, related to that example, is rescaled by the loss. Moreover, the slack-rescaling approach still allows the sum $\sum \xi_i$ to be interpreted as an upper bound on the empirical risk. The slack-rescaling variant is equivalent to applying the augmented hinge loss ℓ^s_Δ depicted in Equation 2.6. Optimization Problem 6.4 with arbitrary loss functions Δ and slack-rescaled losses can be stated as follows.

Optimization Problem 6.4 (SVM, Slack-rescaled Loss) *Given labeled examples, loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+_0$, joint feature mapping Φ, tradeoff $C > 0$, and $r = 1, 2$; the primal SVM optimization problem with slack-rescaling loss is defined as*

$$\min_{\mathbf{w},\boldsymbol{\xi}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{r}\sum_{i=1}^{n} \xi_i^r$$

subject to the constraints

$$\forall_{i=1}^{n}, \forall_{\bar{y}\neq y_i} \ \langle \mathbf{w}, \Phi(\mathcal{x}_i, y_i) - \Phi(\mathcal{x}_i, \bar{y})\rangle \geq 1 - \frac{\xi_i}{\sqrt[r]{\Delta(y_i, \bar{y})}}$$
$$\forall_{i=1}^{n} \ \xi_i \geq 0.$$

Tsochantaridis et al. (2005) derive corresponding 1- and 2-norm dual optimization problems and propose an iterative optimization algorithm that is proven to converge to the optimal solution in polynomial time. Notice, if Δ is the 0/1 loss, Optimization Problems 6.2, 6.3, and 6.4 are equivalent.

6.3 Co-support Vector Machines

We introduced the V-learning objective as the joint minimization of the V regularized empirical risks in addition to the disagreement of all pairs of views. The objective function of SVMs already minimizes an upper bound on the regularized empirical risk. In the remainder of this chapter we focus on two-view learning although all results are easily generalizable to arbitrary V-view learning. To avoid cluttering the notation, we will also focus on the optimization of one view; the superscript $\bar{v}$ indicates variables of the respective peer view.

Thus, in the co-learning setting that we discuss here we have n labeled examples $(\mathcal{x}_1, y_1), \ldots, (\mathcal{x}_n, y_n)$ and m unlabeled inputs $\mathcal{x}_{n+1}, \ldots, \mathcal{x}_{n+m}$. For two-view learning, the joint decision function is given by the sum

$$f(\mathcal{x}, y) = f^1(\mathcal{x}, y) + f^2(\mathcal{x}, y),$$

where each view $f^v(\mathcal{x}, y) = \langle \mathbf{w}^v, \Phi^v(\mathcal{x}, y)\rangle$, $v = 1, 2$ has its respective feature map. According to the consensus maximizing principle, the co-support vector machine now has to minimize the number of errors for labeled examples and the disagreement for the unlabeled examples. For view $v = 1, 2$ this is the case if the labeled examples fulfill Equation 6.2, while for the unlabeled examples

$$f^v(\mathcal{x}_i, \hat{y}_i^{\bar{v}}) - \max_{\bar{\mathbf{y}}\mathbf{y}\neq\hat{y}_i^{\bar{v}}} f^v(\mathcal{x}_i, \bar{y}) = \gamma_i^v \geq 1 \tag{6.3}$$

holds for all $i = n+1, \ldots, n+m$. We call γ_i^v the margin realized by the i-th example in view v. The structure $\hat{y}_i^{\bar{v}}$ denotes the prediction of the peer view $\bar{v}$ that is treated as correct output for the i-th unlabeled input example.

The previous section showed that margin violations induced by a high loss should be penalized more severely than a violation caused by an output that is close to the true target. This can be done by either multiplying the

Table 6.2: *CoSVM Optimization Algorithm with Slack-rescaling*

Input: i-th unlabeled example x_i, $S^1_{j\neq i}$, $S^2_{j\neq i}$, C, C_u, r, repetitions r_{max}.

Initialize $S^1_i = S^2_i = \emptyset$, $\alpha^1_{i\hat{y}^\star_i\bar{y}} = \alpha^2_{i\hat{y}^\star_i\bar{y}} = 0$ **for** all $\bar{y} \in \mathcal{Y}(x_i)$
repeat
  for each view $v = 1, 2$
    $\hat{y}^v_i = \text{argmax}_{\bar{y}} \langle \mathbf{w}^v, \Phi^v(x_i, \bar{y}) \rangle$
    $\bar{y}^v_i = \text{argmax}_{\bar{y}\neq\hat{y}^v_i} (1 - \langle \mathbf{w}^v, \Phi^v_{i,\hat{y}^v_i,\bar{y}} \rangle) \sqrt[r]{\Delta(\hat{y}^v_i, \bar{y})}$
    $\xi^v_i = \max_{\bar{y}\in S^v_i} \{(1 - \langle \mathbf{w}, \Phi^v_{i,\hat{y}^v_i,\bar{y}} \rangle) \sqrt[r]{\Delta(\hat{y}^v_i, \bar{y})}\}$
    $\gamma^v_i = f^v(x_i, \hat{y}^v_i) - f^v(x_i, \bar{y}^v_i)$
  end
  if $\hat{y}^1_i \neq \hat{y}^2_i \vee \langle \mathbf{w}^v, \Phi^v_{i,\hat{y}^v_i,\bar{y}^v_i} \rangle < 1 - \frac{\xi^v_i}{\sqrt[r]{\Delta(\hat{y}^v_i, \bar{y}^v_i)}}$, $v = 1, 2$
    for each view $v = 1, 2$
      Substitute former target $y^v_i = \hat{y}^{\bar{v}}_i$
      if $[[\hat{y}^1 \neq \hat{y}^2]]$
        $S^v_i = S^v_i \cup \{\hat{y}^v_i\}$
      else
        $S^v_i = S^v_i \cup \{\bar{y}^v_i\}$
      end
      Optimize $\alpha^v_{i\hat{y}^{\bar{v}}_i\bar{y}}$ over all $\bar{y} \in S^v_i$ **with** $S^v_{j\neq i}$ fixed
      $\forall \bar{y} \in S^v$ **with** $\alpha^v_{i\hat{y}^{\bar{v}}_i\bar{y}} = 0$: $S^v_i = S^v_i \backslash \{\bar{y}\}$
    end
  end
until consensus **or** r_{max} repetitions

Output: Optimized α^1_i and α^2_i, sets S^1_i and S^2_i.

functional margin by the actual loss or equivalently by rescaling the slack variables with the inverse loss. In the remainder of this section, we follow the latter approach and present co-support vector learning with slack-rescaled losses in greater detail. Nevertheless, all results can easily be obtained for a margin-rescaling approach. We begin with the primal optimization problem and derive the corresponding duals in a subsequent step.

Optimization Problem 6.5 (Primal CoSVM, Slack-rescaling) *Given n labeled examples and m unlabeled examples, loss function Δ, joint feature mappings Φ^1 and Φ^2, let $C, C_u > 0$, $r = 1, 2$; the primal support vector*

optimization problem with slack-rescaling loss in view v is defined as

$$\min_{\mathbf{w}^v, \boldsymbol{\xi}^v} \quad \frac{1}{2}\|\mathbf{w}^v\|^2 + \frac{C}{r}\left(\sum_{i=1}^{n}(\xi_i^v)^r + C_u \sum_{i=n+1}^{n+m}(\min\{\gamma_i^{\bar{v}}, 1\})(\xi_i^v)^r\right)$$

subject to the constraints

$$\forall_{i=1}^{n}, \forall_{\bar{y}\neq y_i} \ \langle \mathbf{w}^v, \Phi^v(\boldsymbol{x}_i, y_i) - \Phi^v(\boldsymbol{x}_i, \bar{y})\rangle \geq 1 - \frac{\xi_i^v}{\sqrt[r]{\Delta(y_i, \bar{y})}}$$

$$\forall_{i=n+1}^{n+m}, \forall_{\bar{y}\neq y^{(\bar{v})}} \ \langle \mathbf{w}^v, \Phi^v(\boldsymbol{x}_i, y_i^{\bar{v}}) - \Phi^v(\boldsymbol{x}_i, \bar{y})\rangle \geq 1 - \frac{\xi_i^v}{\sqrt[r]{\Delta(y_i^{\bar{v}}, \bar{y})}}$$

$$\forall_{i=1}^{n+m} \ \xi_i^v \geq 0,$$

where the superscript $\bar{v}$ denotes variables from the peer view.

The balancing factor C_u regularizes the influence of the unlabeled data. Weights of $\min\{\gamma_i^{\bar{v}}, 1\}$ to the slacks $\xi_{n+1}^v, \ldots, \xi_{n+m}^v$ relate errors on unlabeled examples to the margin of the peer view's prediction. Thus, an unlabeled sequence satisfying the margin constraint has the same influence in the peer view as the labeled examples. The sum of the slack variables now consists of an upper bound on the error for the labeled examples and an upper bound on the disagreement weighted by the confidence of the peer view's prediction.

Similarly to the regular support vector machine, the constraints of Optimization Problem 6.5 can be integrated into the objective function by introducing non-negative Lagrange multipiers $\alpha_{i\hat{y}_i^\star\bar{y}}^v$ and β_i^v for all $i = 1, \ldots, n+m$ and every $\bar{y} \in \mathcal{Y}(\boldsymbol{x}_i)$. For $r = 1$, the corresponding Lagrangian of Optimization Problem 6.5 in view $v = 1, 2$ takes the form

$$\begin{aligned} Q(\mathbf{w}^v, \boldsymbol{\xi}^v) =& \frac{1}{2}\|\mathbf{w}^v\|^2 + C\left(\sum_{i=1}^{n}\xi_i^v + C_u\sum_{i=n+1}^{n+m}(\min\{\gamma_i^{\bar{v}}, 1\})\xi_i^v\right) - \sum_{i=1}^{n+m}\beta_i^v\xi_i^v \\ & - \sum_{i=1}^{n}\sum_{\substack{\bar{y}\in\mathcal{Y}(\boldsymbol{x}_i)\\ \bar{y}\neq\hat{y}_i^\star}} \alpha_{i\hat{y}_i^\star\bar{y}}^v\Big(\langle \mathbf{w}^v, \Phi^v(\boldsymbol{x}_i, \hat{y}_i^\star) - \Phi^v(\boldsymbol{x}_i, \bar{y})\rangle - 1 + \frac{\xi_i^v}{\Delta(\hat{y}_i^\star, \bar{y})}\Big), \end{aligned}$$

where $\hat{y}_i^\star$ denotes either the true output for labeled examples, or equals the prediction of the peer view for unlabeled examples, that is,

$$\hat{y}_i^\star = \begin{cases} y_i & : \ 1 \leq i \leq n \\ \operatorname{argmax}_{\bar{y}} f^{\bar{v}}(\boldsymbol{x}_i, \bar{y}) & : \ n+1 \leq i \leq n+m. \end{cases} \tag{6.4}$$

Treating the predictions of the peer view as constant, the Lagrangian of view v is convex and the Karush Kuhn Tucker (KKT) conditions, that are

necessary and sufficient for $\mathbf{w}$ and $\boldsymbol{\xi}$ to be a solution, hold. Differentiating the Lagrangian with respect to the slack variables leads to the following equations for labeled examples,

$$\frac{\partial Q}{\partial \xi_i} = C - \sum_{\substack{\bar{y}\in\mathcal{Y}(x_i)\\ \bar{y}\neq y_i}} \frac{\alpha^v_{iy_i\bar{y}}}{\Delta(y_i,\bar{y})} - \beta^v_i \quad \stackrel{!}{=} \quad 0,$$

and unlabeled examples,

$$\frac{\partial Q}{\partial \xi_j} = (\min\{\gamma^{\bar{v}}_j, 1\})C_u C - \sum_{\substack{\bar{y}\in\mathcal{Y}(x_j)\\ \bar{y}\neq \operatorname{argmax}_{\bar{y}'} f^{\bar{v}}(x_j,\bar{y}')}} \frac{\alpha^v_{j,\operatorname{argmax}_{\bar{y}'} f^{\bar{v}}(x_j,\bar{y}'),\bar{y}}}{\Delta\big(\operatorname{argmax}_{\bar{y}'} f^{\bar{v}}(x_j,\bar{y}'),\bar{y}\big)} - \beta^v_j \quad \stackrel{!}{=} \quad 0,$$

respectively. Using the nonnegativity of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ leads to the generalized box-constraints. We obtain

$$\forall_{i=1}^{n} \qquad \sum_{\substack{\bar{y}\in\mathcal{Y}(x_i)\\ \bar{y}\neq y_i}} \frac{\alpha^v_{iy_i\bar{y}}}{\Delta(y_i,\bar{y})} \leq C \tag{6.5}$$

$$\forall_{j=n+1}^{n+m} \qquad \sum_{\substack{\bar{y}\in\mathcal{Y}(x_j)\\ \bar{y}\neq \hat{y}^{\bar{v}}_j}} \frac{\alpha^v_{j\hat{y}^{\bar{v}}_j\bar{y}}}{\Delta\big(\hat{y}^{\bar{v}}_j,\bar{y}\big)} \leq (\min\{\gamma^{\bar{v}}_j, 1\})C_u\, C, \tag{6.6}$$

where $\hat{y}^{\bar{v}}_j = \operatorname{argmax}_{\bar{y}'} f^{\bar{v}}(x_j,\bar{y}')$ denotes the prediction of the peer view for the j-th unlabeled example. Taking the derivative of the remaining part of the Lagrangian with respect to the weight vector $\mathbf{w}$ gives us

$$\frac{\partial Q}{\partial \mathbf{w}^v} = \mathbf{w}^v - \sum_{i=1}^{n+m} \sum_{\substack{\bar{y}\in\mathcal{Y}(x_i)\\ \bar{y}\neq \hat{y}^\star_i}} \alpha^v_{i\hat{y}^\star_i\bar{y}} \Big(\Phi^v(x_i,\hat{y}^\star_i) - \Phi^v(x_i,\bar{y})\Big) \stackrel{!}{=} 0.$$

Introducing the shorthand notation for difference vectors $\Phi_{i\hat{y}^\star_i\bar{y}} = \Phi(x_i,\hat{y}^\star_i) - \Phi(x_i,\bar{y})$, the weight vector can be represented in terms of the Lagrange multipliers $\alpha_{i\hat{y}^\star_i\bar{y}}$; we have

$$\begin{aligned}\mathbf{w}^v &= \sum_{i=1}^{n} \sum_{\substack{\bar{y}\in\mathcal{Y}(x_i)\\ \bar{y}\neq y_i}} \alpha^v_{iy_i\bar{y}} \Phi^v_{i,y_i,\bar{y}} + \sum_{j=n+1}^{n+m} \sum_{\substack{\bar{y}\in\mathcal{Y}(x_j)\\ \bar{y}\neq \hat{y}^{\bar{v}}_j}} \alpha^v_{j\hat{y}^{\bar{v}}_j\bar{y}} \Phi^v_{j,\hat{y}^{\bar{v}}_j,\bar{y}} \\ &= \sum_{i=1}^{n+m} \sum_{\substack{\bar{y}\in\mathcal{Y}(x_i)\\ \bar{y}\neq \hat{y}^\star_i}} \alpha^v_{i\hat{y}^\star_i\bar{y}} \Phi^v_{i,\hat{y}^\star_i,\bar{y}}. \end{aligned} \tag{6.7}$$

Substituting Equation 6.7 and constraints 6.5 and 6.6 into the Lagrangian removes its dependence on the primal variables and we resolve the corresponding dual optimization problem that has to be maximized with respect to the $\alpha_{i\hat{y}_i\bar{y}}$. Before stating the dual, we need to define the composite kernel $k\left(x_i, \bar{y}, x_j, \bar{y}'\right)$ that computes the inner product of two difference vectors in input-output space and is given by

$$\begin{aligned} k^v\left(x_i, \hat{y}_i^\star, \bar{y}, x_j, \hat{y}_j^\star, \bar{y}'\right) &= \langle \Phi^v(x_i, \hat{y}_i^\star), \Phi^v(x_j, \hat{y}_j^\star)\rangle - \langle \Phi^v(x_i, \hat{y}_i^\star), \Phi^v(x_j, \bar{y}')\rangle \\ &\quad - \langle \Phi^v(x_i, \bar{y}), \Phi^v(x_j, \hat{y}_j^\star)\rangle + \langle \Phi^v(x_i, \bar{y}), \Phi^v(x_j, \bar{y}')\rangle \\ &= \langle \Phi^v(x_i, \hat{y}_i^\star) - \Phi^v(x_i, \bar{y})\,,\, \Phi^v(x_j, \hat{y}_j^\star) - \Phi^v(x_j, \bar{y}')\rangle \\ &= \langle \Phi^v_{i,\hat{y}_i^\star,\bar{y}}, \Phi^v_{j,\hat{y}_j^\star,\bar{y}'}\rangle. \end{aligned}$$

Putting everything together, we derive the 1-norm co-support vector machine optimization problem with slack-rescaling loss.

Optimization Problem 6.6 (L_1 Dual CoSVM with Slack-rescaling) *Given n labeled and m unlabeled examples, composite kernel k^1 and k^2, and loss function Δ, $C, C_u > 0$; the 1-norm dual co-support vector optimization problem with slack-rescaling loss in view $v = 1, 2$ is defined as*

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n+m} \sum_{\substack{\bar{y}\in\mathcal{Y}(x_i)\\ \bar{y}\neq\hat{y}_i^\star}} \alpha^v_{i\hat{y}_i^\star\bar{y}} - \frac{1}{2}\sum_{i,j=1}^{n+m} \sum_{\substack{\bar{y}\neq\hat{y}_i^\star\\ \bar{y}'\neq\hat{y}_j^\star}} \alpha^v_{i\hat{y}_i^\star\bar{y}}\,\alpha^v_{j\hat{y}_j^\star\bar{y}'}\, k^v\big(x_i, \hat{y}_i^\star, \bar{y}, x_j, \hat{y}_j^\star, \bar{y}'\big)$$

subject to the constraints

$$\begin{aligned} \forall_{i=1}^{n} \sum_{\substack{\bar{y}\in\mathcal{Y}(x_i)\\ \bar{y}\neq y_i}} \frac{\alpha^v_{iy_i\bar{y}}}{\Delta(y_i, \bar{y})} &\leq C \\ \forall_{i=n+1}^{n+m} \sum_{\substack{\bar{y}\in\mathcal{Y}(x_i)\\ \bar{y}\neq\hat{y}_i^{\bar{v}}}} \frac{\alpha^v_{i\hat{y}_i^{\bar{v}}\bar{y}}}{\Delta(\hat{y}_i^{\bar{v}}, \bar{y})} &\leq (\min\{\gamma_i^{\bar{v}}, 1\})C_u\, C \\ \forall_{i=1}^{n+m} \forall_{\substack{\bar{y}\in\mathcal{Y}(x_i)\\ \bar{y}\neq\hat{y}_i^\star}} \alpha_i^v(\hat{y}_i^\star, \bar{y}) &\geq 0. \end{aligned}$$

Applying the L_2-norm, Optimization Problem 6.5 is particularly valid for negative values of ξ_i since $\xi_i^v < 0$ satisfies the constraints

$$\langle \mathbf{w}^v, \Phi^v_{i,\hat{y}_i^\star,\bar{y}}\rangle \geq 1 - \frac{\xi_i^v}{\sqrt{\Delta(\hat{y}_i^\star, \bar{y})}} \tag{6.8}$$

for every i and $\bar{y} \in \mathcal{Y}(x_i)$ and moreover the sum $\sum_i (\xi_i^v)^2$ guarantees the objective to be positive. We thus may drop the non-negativity constraint on the slack variables $\xi_i^v \geq 0$. Setting $r = 2$, the dual 2-norm co-support vector machine optimization problem for slack-rescaling loss can be derived analogously to the 1-norm case by re-substituting primal derivatives with respect to $\mathbf{w}^v$ and ξ_i^v into the Lagrangian. Similar to the regular 2-norm support vector machine, additional constraints are augmented into the kernel function. The derivation of the corresponding 2-norm dual is straight forward; we will directly state the optimization problem.

Optimization Problem 6.7 (L_2 Dual CoSVM with Slack-rescaling) *Given n labeled and m unlabeled examples, composite kernels k^1 and k^2, and loss function Δ, $C, C_u > 0$; the 2-norm dual co-support vector optimization problem with slack-rescaling loss in view $v = 1, 2$ is defined as*

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n+m} \sum_{\substack{\bar{y} \in \mathcal{Y}(x_i) \\ \bar{y} \neq \hat{y}_i^\star}} \alpha_{i\hat{y}_i^\star \bar{y}}^v - \frac{1}{2} \sum_{i,j=1}^{n+m} \sum_{\substack{\bar{y} \neq \hat{y}_i^\star \\ \bar{y}' \neq \hat{y}_j^\star}} \alpha_{i\hat{y}_i^\star \bar{y}}^v \, \alpha_{j\hat{y}_j^\star \bar{y}'}^v \, k_2^v\big(x_i, \hat{y}_i^\star, \bar{y}, x_j, \hat{y}_j^\star, \bar{y}'\big)$$

subject to the constraints $\forall_{v=1}^{2} \forall_{i=1}^{n+m} \forall_{\bar{y} \neq \hat{y}_i^\star} \, \alpha_{i\hat{y}_i^\star \bar{y}}^v \geq 0$, with composite kernel $k_2^v(x_i, \hat{y}_i^\star, \bar{y}, x_j, \hat{y}_j^\star, \bar{y}') = k^v(x_i, \hat{y}_i^\star, \bar{y}, x_j, \hat{y}_j^\star, \bar{y}') + \delta'_{i,\hat{y}_i^\star,\bar{y},j,\hat{y}_j^\star,\bar{y}'}$ where

$$\delta'_{i,\hat{y}_i^\star,\bar{y},j,\hat{y}_j^\star,\bar{y}'} = \begin{cases} (C\sqrt{\Delta(y_i,\bar{y})\Delta(y_j,\bar{y}')})^{-1} & : \quad i = j \leq n \\ \Big((\min\{\gamma_j^{\bar{v}}, 1\}) C_u \, C \sqrt{\Delta(\hat{y}_i^{\bar{v}},\bar{y})\Delta(\hat{y}_j^{\bar{v}},\bar{y}')}\Big)^{-1} & : \quad n < i = j \\ 0 & : \quad \textit{otherwise.} \end{cases}$$

6.4 Optimization Strategy

The optimization of the dual problems of the previous section is difficult since either view depends on predictions of the peer view for unlabeled examples. These predictions may change during the training process and thus turn the problem into a nonconvex one. We implicitly deal with this by applying a cutting plane method (Kelley, 1960). In our setting, a cutting plane translates into finding the most strongly violated constraint and adding it to the working set if the current violation is larger than those realized by constraints already in the working set. In doing so, we successively shrink the feasible region in the primal, or equivalently, instantiate a new dual variable, associated with this constraint, see also (Altun et al., 2007).

Since the dual variables $\alpha_{iy_i^\star \bar{y}}^v$ are tied to observations x_i, the dual optimization problem splits into $n + m$ disjoint subspaces spanned by every

α_i with fixed values for the $\alpha_{j \neq i}$ (Joachims, 1999b); the optimization iterates over these subspaces as follows. In an outer loop, the co-support vector machine iterates over the examples and consecutively optimizes the associated parameters α_i, using distinct working set approaches for labeled (Table 6.1, see also Tsochantaridis et al. 2005) and unlabeled (Table 6.2) examples, adding a new constraint in each iteration if necessary.

Labeled instances are optimized in either view $v = 1, 2$ separately. Algorithm 6.1 computes the most strongly violated constraint $\bar{\boldsymbol{y}}_i^v$ in view v (line 3) and relates the induced error to the actual slack for that example (line 4). If the margin constraint is violated (line 5), an update has to be performed. That is, the erroneously decoded output is added to the working set (line 6) and the dual parameters are optimized over the subspace spanned by the i-th input example (line 7). For an unlabeled instance, Algorithm 6.2 computes the top scoring output (line 4) and its best runner-up (line 5) for both views and relates their difference to the current slack (line 6). The if-clause in line 9 checks whether both views disagree on the prediction and determines possible margin violations in the views. If an error is detected, an update is performed (line 10-18).

The proposed optimization scheme leads to sparse models, since it suffices to explicitly store only those $\alpha_{i\hat{\boldsymbol{y}}_i^\star \bar{\boldsymbol{y}}}$ whose associated output $\bar{\boldsymbol{y}}$ is decoded instead of the true $\boldsymbol{y}_i$ or the prediction of the peer view $\hat{\boldsymbol{y}}_i^{\bar{v}}$, respectively. All dual variables that are not included in the active working set are implicitly considered to be zero. Outputs $\bar{\boldsymbol{y}}$ with $\alpha_{i\hat{\boldsymbol{y}}_i^\star \bar{\boldsymbol{y}}} = 0$ are removed from the working set in order to speed up computation. When the loop reaches an example for the second time, all former outputs $\alpha_{i\hat{\boldsymbol{y}}_i^\star \bar{\boldsymbol{y}}}$ of that example are removed since the errors or disagreements that they used to correct in earlier iterations of the main loop may have been resolved. Since the cost factors upper bound the growth of the $\alpha_{i\hat{\boldsymbol{y}}_i^\star \bar{\boldsymbol{y}}}$, consensus might not be established and we therefore integrate a user defined constant r_{max} that bounds the number of iterations.

6.5 Empirical Results

We investigate our approach by applying the semi-supervised support vector machine to multi-class classification (Section 6.5.1), named entity recognition (Section 6.5.2), and natural language parsing (Section 6.5.3). We explore the benefits of co-learning and investigate its execution time. The baseline SVM is described by Tsochantaridis et al. (2005).

In each setting, the influence of unlabeled examples is determined by a smoothing strategy which exponentially approaches C_u after a fixed number of epochs. We first optimize parameter C_u using resampling; we then fix C_u

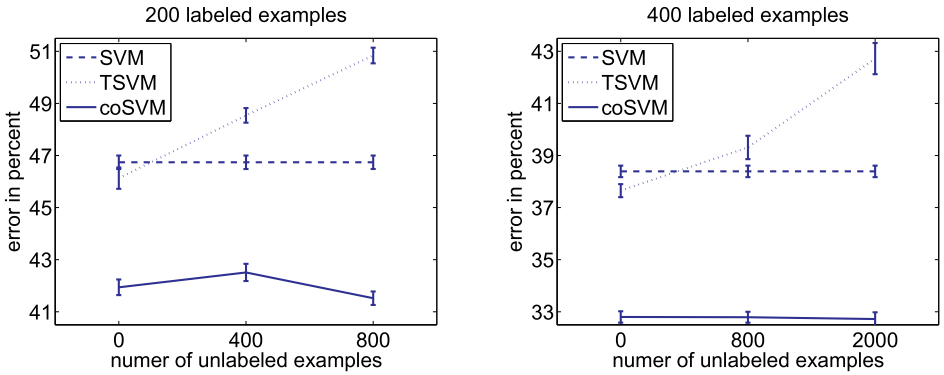


Figure 6.1: *Error rates for the Cora data set for 200 (left) and 400 (right) labeled examples and varying numbers of unlabeled examples. Displayed are structured SVM (dashed), transductive SVM (dotted), and coSVM (solid).*

and present curves that show the average error over distinct randomly drawn training and holdout sets. The baseline methods SVM and also TSVM^{light} for the multi-class classification and structural perceptron for the NER experiments, are trained on concatenated views. We initialize $r_{max} = 10$, $C = 1$. For all problems and sample sizes we conduct a one-sided t-test at a 1% confidence level. Significant results are indicated in the text.

6.5.1 Multi-Class Classification

Our multi-class classification experiments are based on the Cora data set that contains 9,947 linked computer science papers. We remove documents without a reference section and obtain 9,555 papers divided into eight different classes. We exploit the link structure and generate two natural views of the documents: a term frequency view of the document and an outlink view. We extract term frequencies of the document and of the anchor text of the inbound links. The latter consists of the term frequencies of three sentences, centered at the occurrence of the reference.

We use the common 0/1 loss and the respective 2-norm variant of both structured prediction methods and the transductive SVM^{light} (Joachims, 1999a), trained with a one-against-all strategy, as an additional baseline. Figure 6.1 details error rates and standard errors in percent for different numbers of labeled and unlabeled training examples and 500 holdout examples. The results are averages over 100 repetitions with distinct training and holdout sets. The performance of the TSVM deteriorates when the number of unlabeled instances is increased. The co-trained SVM significantly outperforms its fully supervised counterpart for all numbers of labeled and

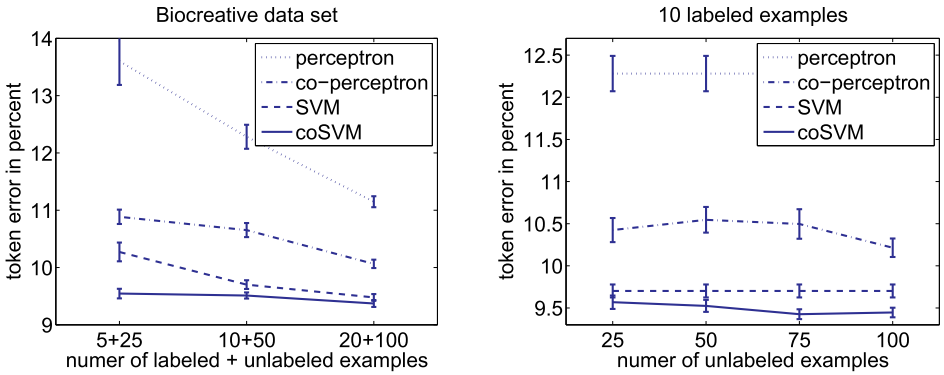


Figure 6.2: *Token error for the Biocreative data set. Displayed are perceptron (dashed-dotted), co-perceptron (dotted), structured SVM (dashed) and coSVM (solid).*

unlabeled examples. However, the number of unlabeled examples has no significant effect on the results.

6.5.2 Named Entity Recognition

We look at the effectiveness of our approach on two named entity recognition problems. We use the data set provided for task 1A of the Biocreative challenge and the Spanish news wire article corpus of the shared task of CoNLL 2002.

The Biocreative data contains 7500 sentences from biomedical papers; gene and protein names are to be recognized. We discriminate tokens that are parts of gene names against all other tokens. We utilize *label-observation* features like the token itself, letter 2,3 and 4-grams, and surface clues like capitalization, inclusion of Greek symbols, numbers, and others. The CoNLL2002 data contains 9 label types which distinguish between person, organization, location, and other names. We use 3100 sentences of between 10 and 40 tokens. The extracted *label-observation* features cover the token itself and surface clues.

We ensure that each label occurs at least once in the labeled training data; otherwise, we discard and draw again. The holdout sets consist of either 500 Biocreative or 300 Spanish news wire sentences, respectively. We utilize a random feature split of the attributes into two views for each repetition. We report token errors in percent for coSVM, single-view SVM, and also for supervised perceptron and co-perceptron (Chapter 5) as an additional baseline for both data sets.

Figure 6.2 (left) shows learning curves for all four algorithms for the

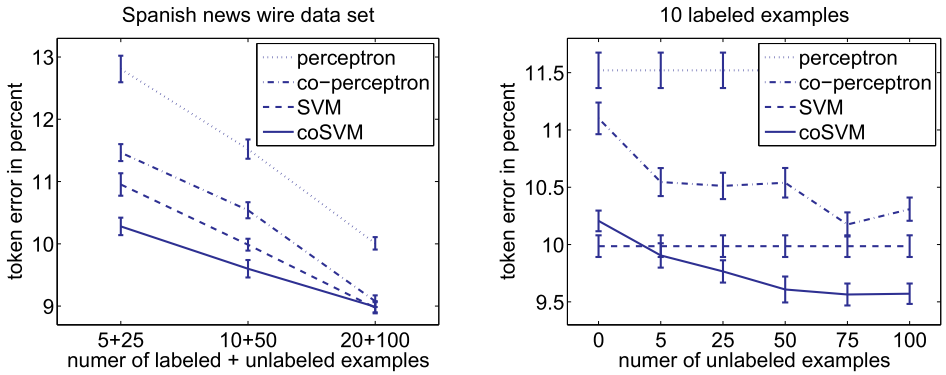


Figure 6.3: *Token error for the Spanish news wire data set. Left: training curves for perceptron (dashed-dotted), co-perceptron (dotted), structured SVM (dashed) and coSVM (solid). Right: error depending on the unlabeled sample size.*

Biocreative data set. The two perceptrons are clearly outperformed by both support vector algorithms. CoSVM leads to significantly lower error rates than its supervised counterpart. When we fix the number of labeled examples and vary the number of unlabeled instances in Figure 6.2 (right) we observe that coSVM effectively utilizes the unlabeled data. The error rates decrease significantly when the number of unlabeled instances increases.

Learning curves for the Spanish news wire data set are detailed in Figure 6.3 (right). Except for 20 labeled and 100 unlabeled instances, the semi-supervised algorithms outperform their fully supervised counterparts significantly. Increasing the number of unlabeled data in Figure 6.3 leads to significantly better error rates for co-perceptron and co-support vector machines.

6.5.3 Natural Language Parsing

For our natural language parsing experiments we learn an unlexicalized, weighted context-free grammar on subsets of the Penn treebank Wall Street Journal corpus (Marcus et al., 1993, 1994) and the Negra corpus (Skut et al., 1997). Both are tagged with part-of-speech and completely annotated with syntactic structures.

We use the subsets 2-21 of the Wall Street Journal corpus that contain 39,833 sentences. We extract sentences of length of at most 15 words. From the annotations of the resulting 8,666 sentences we build a context-free grammar of approximately 4,800 distinct production rules. The Negra corpus contains 20,602 sentences from a German newspaper archive. We extract sentences of between 5 and 25 tokens. The resulting 14,137 sentences contain

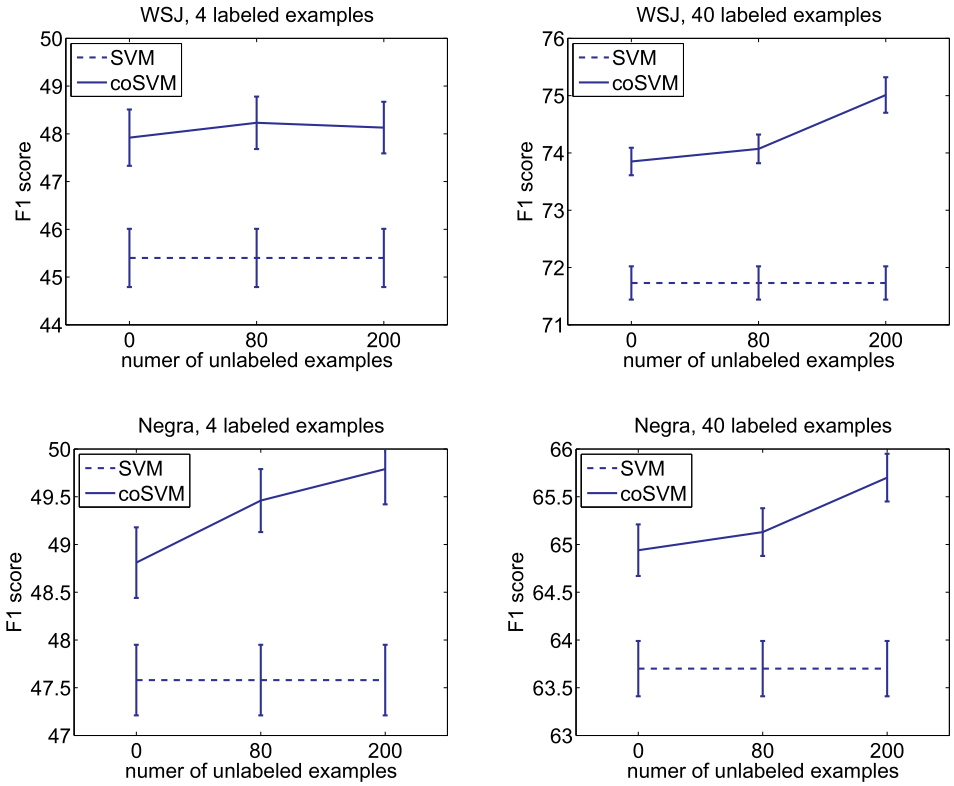


Figure 6.4: *F1 scores for the Wall Street Journal (WSJ) corpus (top) and the Negra corpus (bottom). Displayed are results for 4 labeled (left column) and 40 labeled (right column) examples and varying numbers of unlabeled examples.*

more than 26,700 production rules in Chomsky normal form.

The extracted local feature maps contain the rule itself and binarized border and span width features for both corpora. The algorithms are evaluated on different numbers of labeled and unlabeled examples. Each result is averaged over 100 repetitions. In each repetition we use distinct, randomly chosen feature splits and randomly drawn training and holdout sets. The latter is of size 100. We use a modified variant of the CKY implementation by Johnson (1998) for the decoding and apply 2-norm SVMs with the loss $\Delta(y_i, \bar{y}) = 1 - F_1(y_i, \bar{y})$ (see also Section 3.2.3).

Figure 6.4 (top row) details F_1 scores for different numbers of labeled and unlabeled training instances for the Wall Street Journal corpus. Surprisingly, even with no unlabeled data, coSVM leads to better F_1 scores than regular SVM by simply averaging the predictions of the two views. When we add unlabeled instances, the performance of coSVM increases. Note that additional unlabeled examples further improve F_1 score significantly.

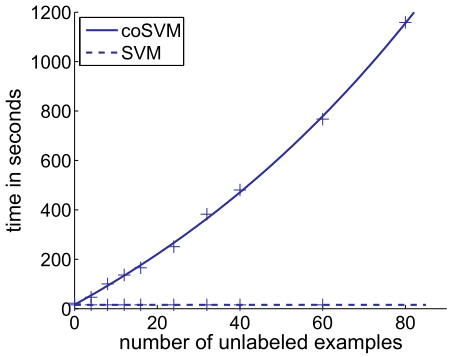


Figure 6.5: *Execution time.*

The results for experiments with the Negra corpus are shown in Figure 6.4 (bottom row). The observations are very similar. Again, averaging the independently trained hypotheses leads to better F_1 scores than regular SVM. The performance is further significantly increased by adding unlabeled data.

6.5.4 Execution Time

The observed performance benefits of coSVM are at the cost of significantly longer training processes. Figure 6.5 plots execution time against training set size for 4 labeled and different numbers of unlabeled examples. Empirically, we observe that the execution time of co-trained SVM scales between linearly and quadratically in the number of unlabeled examples.

6.6 Conclusions

We devised a semi-supervised variant of the support vector machine for structured output variables and arbitrary loss functions (coSVM). It is based on the co-training framework and implements the principle of consensus maximization between hypotheses. We derived 1- and 2-norm optimization problems that not only allow for the usage of arbitrary loss functions but also allow for the use of arbitrary feature mappings and corresponding decoding strategies.

We used various norms, loss functions, and feature splits in our experiments. Empirical results for multi-class classification, named entity recognition, and natural language parsing tasks showed that coSVM leads to better models in terms of the chosen loss function compared to the fully-supervised SVM. We observe that the co-trained support vector machine with no unla-

beled examples significantly outperforms the baseline methods in all tasks. We credit this finding to averaging two independently trained hypotheses. However, the prediction accuracy of coSVM can be further increased by adding unlabeled examples in the named entity recognition and parsing experiments.

The increase in performance comes at the cost of longer execution times. The semi-supervised support vector machine benefits from the inclusion of unlabeled examples into the training process. We observed that coSVM significantly outperforms its single-view counterpart in all tasks.

The presented coSVM generalizes several lines of research, such as binary co-training approaches (Blum and Mitchell, 1998; Brefeld and Scheffer, 2004) structured perceptrons (Collins and Duffy, 2002; Altun et al., 2003b), maximum Markov networks (Taskar et al., 2004a), and SVMs for structured output spaces (Tsochantaridis et al., 2005).

Notice that this optimization scheme is independent of the joint feature mapping Φ^1 and Φ^2 (either explicitly or via a kernel function) and the loss function Δ. Due to this general setup, the coSVM can be applied to a variety of problems. To apply coSVM, it suffices to define appropriate joint feature mappings, a loss function, and a corresponding decoding for the problem at hand.

Chapter 7

Transductive Support Vector Machines

In this chapter, we study the problem of learning kernel machines transductively for structured output variables. Transductive learning can be reduced to combinatorial optimization problems over all possible labelings of the unlabeled data. In order to scale combinatorial semi-supervised learning to structured variables, we transform the corresponding non-convex, combinatorial, constrained optimization problems into continuous, unconstrained optimization problems. The discrete optimization parameters are eliminated and the resulting differentiable problems can be optimized efficiently.

Several semi-supervised techniques in joint input-output spaces have been studied. One of the most promising approaches is the integration of unlabeled instances by Laplacian priors into structured large margin classifiers (Lafferty et al., 2004; Altun et al., 2006). Lee et al. (2007) study semi-supervised CRFs and include unlabeled data via an entropy criterion such that their objective acts as a probabilistic analogon to the transductive setting we discuss here. Xu et al. (2006) derive unsupervised M^3Networks by employing SDP relaxation techniques. Their optimization problem is similar to the transductive criterion derived in this paper.

The technique of binary unconstrained gradient-based optimization of SVMs (Chapelle, 2006) is based on the observation that constraints of the form $y_i\langle \mathbf{w}, \mathbf{x}_i\rangle \geq 1-\xi_i$ can be written as equalities $\xi_i = \max\{1-y_i\langle \mathbf{w}, \mathbf{x}_i\rangle, 0\}$. Inserting these expressions of ξ_i into the primal SVM criterion leads to an unconstrained, convex quadratic optimization problem. When the maximum is replaced by a softmax, the resulting differentiable criterion can be minimized, for instance, by conjugate gradient descent. This technique unfolds its benefit for semi-supervised learning based on the TSVM criterion (Chapelle and Zien, 2005). Traditional TSVM implementations (Ben-

net and Demiriz, 1998; Joachims, 1999a) solve a combinatorial optimization problem with pseudo-labels $\hat{y}_j$ for the unlabeled $\mathbf{x}_j$. These additional discrete optimization parameters can be removed from the problem altogether when the constraints $\hat{y}_j\langle \mathbf{w}, \mathbf{x}_i\rangle \geq 1-\xi_i$ are expressed using absolute values: $\xi_j = \max\{1 - |\langle \mathbf{w}, \mathbf{x}_j\rangle|, 0\}$. The resulting problem remains non-convex, but is now continuous and has fewer parameters.

When unlabeled data and two distinct views Φ^1 and Φ^2 on the instances are available, co-learning can be employed. Co-learning algorithms employ a mechanism that reduces the rate of disagreement on unlabeled data between two hypotheses, thus reducing an upper bound on their error rate. This disagreement minimization, has been introduced in previous chapters by iterative label-exchanging strategies. CoSVM has two optimization criteria that are interleaved by exchanging labelings for unlabeled inputs. We present an optimization problem that integrates them into a joint criterion.

The structure of this chapter and its main contributions are as follows. We leverage the technique of continuous optimization to structured input-output spaces, addressing the supervised (Section 7.1), the transductive (Section 7.2), and the multi-view case (Section 7.3). Our treatment covers general loss functions and linear discrimination as well as general kernels. We empirically examine the benefit of generalized transductive SVMs for multi-class classification and label sequence learning tasks in Section 7.4. Section 7.5 provides a discussion of the experimental results and Section 7.7 concludes.

7.1 Unconstrained Optimization for Structured Output Spaces

Optimization Problem 7.1 is the known SVM learning problem in input-output spaces with cost-based margin-rescaling, which includes the fixed size margin with 0/1-loss as a special case. All presented results can also be derived for slack-rescaling approaches; however, the corresponding constraint generation becomes more difficult. Generally, the norm of $\mathbf{w}$ plus the sum of the slack terms ξ_i is minimized, subject to the constraint that, for all examples $(\mathcal{x}_i, \mathcal{y}_i)$, the correct label $\mathcal{y}_i$ receives the highest score by a margin. Using the shorthand $\Phi_{i\mathcal{y}_i\bar{\mathcal{y}}} = \Phi(\mathcal{x}_i, \mathcal{y}_i) - \Phi(\mathcal{x}_i, \bar{\mathcal{y}})$, we can state the optimization problem of the structural support vector machine as follows.

Optimization Problem 7.1 (SVM) *Given n labeled training pairs, loss function Δ and $C > 0$, the primal constraint SVM optimization problem is*

defined as

$$\begin{array}{lll} \min\limits_{\mathbf{w},\boldsymbol{\xi}} & ||\mathbf{w}||^2 + C\sum\limits_{i=1}^{n}\xi_i & \\ s.t. & \forall_{i=1}^{n}\,\forall_{\substack{\bar{y}\in\mathcal{Y}(x_i)\\ \bar{y}\neq y_i}}\ \langle \mathbf{w}, \Phi_{iy_i\bar{y}}\rangle & \geq \Delta(y_i,\bar{y}) - \xi_i, \\ & \forall_{i=1}^{n}\ \xi_i & \geq 0. \end{array}$$

In general, unconstrained optimization is easier to implement than constrained optimization. For the SVM, it is possible to resolve the slack terms:

$$\begin{aligned} \xi_i &= \max\left\{\max_{\bar{y}\neq y_i}\left\{\Delta(y_i,\bar{y}) - \langle \mathbf{w}, \Phi_{iy_i\bar{y}}\rangle\right\}, 0\right\} \\ &= \max\nolimits_{\bar{y}\neq y_i}\left\{\ell_{\Delta(y_i,\bar{y})}\left(\langle \mathbf{w}, \Phi_{iy_i\bar{y}}\rangle\right)\right\}, \end{aligned} \tag{7.1}$$

where $\ell_\Delta(t) = \max\{\Delta - t, 0\}$ is the hinge loss with margin-rescaling (see Equation 2.5). We can now pose Optimization Problem 7.2 for structured outputs, a simple quadratic optimization function *without constraints*.

Optimization Problem 7.2 (∇SVM) *Given n labeled training pairs, loss function Δ, and let $C > 0$; the unconstrained SVM optimization problem is defined as*

$$\min_{\mathbf{w}} \quad ||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i, \tag{7.2}$$

where $\xi_i = \max\nolimits_{\bar{y}\neq y_i}\ell_{\Delta(y_i,\bar{y})}\left(\langle \mathbf{w}, \Phi_{iy_i\bar{y}}\rangle\right)$.

The ξ_i remain in Optimization Problem 7.2 for better comprehensibility; when they are expanded, the criterion is a closed expression. When the maximum is approximated by the softmax, and a smooth approximation of the hinge loss is used, Equation 7.1 is also differentiable. Following the latter, we substitute the Huber loss for the hinge loss. The differentiable surrogat is based on a quadratic approximation around zero and equals the hinge loss otherwise (see Figure 7.1). The Huber loss $\ell_{\Delta,\epsilon}$ and its first derivative is given by

$$\ell_{\Delta,\epsilon}(t) = \begin{cases} \Delta - t & : \ t \leq \Delta - \epsilon \\ \frac{(\Delta+\epsilon-t)^2}{4\epsilon} & : \ \Delta - \epsilon \leq t \leq \Delta + \epsilon \\ 0 & : \ \text{otherwise} \end{cases}$$

$$\frac{\partial \ell_{\Delta,\epsilon}}{\partial t} = \begin{cases} -1 & : \ t \leq \Delta - \epsilon \\ -\frac{1}{2}(\frac{\Delta-t}{\epsilon} + 1) & : \ \Delta - \epsilon \leq t \leq \Delta + \epsilon \\ 0 & : \ \text{otherwise}\ . \end{cases}$$

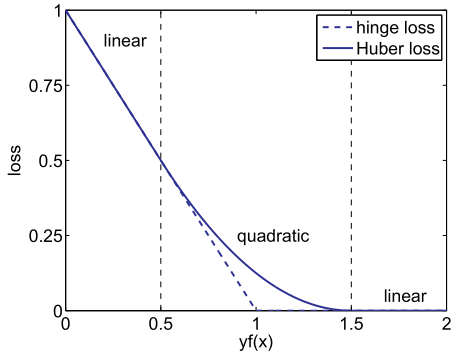


Figure 7.1: *The differentiable Huber loss* $\ell_{\Delta=1,\epsilon=0.5}$.

The softmax can be derived by using the ρ-norm of a vector $\mathbf{z} = (z_1, \ldots, z_d)^\mathsf{T}$,

$$\|\mathbf{z}\|_\rho = \sqrt[\rho]{\sum_{i=1}^{d} z_i^\rho}$$

that is known to converge to the maximal element z_i when $\rho \to \infty$. To derive a differentiable approximation we apply the exponential operator to $\mathbf{z}$ in a component-wise manner by defining

$$\|\exp\{\mathbf{z}\}\|_\rho \stackrel{def}{=} \sqrt[\rho]{\sum_{i=1}^{d} \big(\exp\{z_i\}\big)^\rho}. \tag{7.3}$$

Taking the natural logarithm of Equation 7.3 leads to the so-called softmax function

$$\log \|\exp\{\mathbf{z}\}\| = \frac{1}{\rho} \log \sum \exp\{\rho |z_i|\}. \tag{7.4}$$

Since all transformations are monotonic, the parameter ρ still controls the degree of approximating the maximum, that is for $\rho \to \infty$ we precisely obtain the maximum operator. We use a slightly different form of the softmax as an approximation of the maximum. First, we extend it to sets instead of vectors and secondly, we incorporate some constants allowing an interpretion for $\rho \to 0$ as the sum of the elements. The softmax and its derivative are displayed in the following equations,

$$\operatorname*{smax}_{\tilde{y} \neq y_k}(s(\tilde{y})) = \frac{1}{\rho} \log \left(1 + \sum\nolimits_{\tilde{y} \neq y_k} (\exp\{\rho s(\tilde{y})\} - 1)\right)$$

$$\frac{\partial}{\partial s(\bar{y})} \operatorname*{smax}_{\tilde{y} \neq y_k}(s(\tilde{y})) = \frac{\exp\{\rho s(\bar{y})\}}{1 + \sum_{\tilde{y} \neq y_k} (\exp\{\rho s(\tilde{y})\} - 1)},$$

where we will use $s(\tilde{y}) = \ell_{\Delta(y_i,\tilde{y})}\left(\langle \mathbf{w}, \Phi_{iy_i\tilde{y}}\rangle\right)$. An application of the representer theorem shows that $\mathbf{w}$ can be expanded as

$$\mathbf{w} = \sum\nolimits_k \sum\nolimits_{\bar{y}\neq y_k} \alpha_{ky_k\bar{y}}\, \Phi_{ky_k\bar{y}}. \tag{7.5}$$

The gradient is a vector over dual parameters $\alpha_{ky_k\bar{y}}$ for each example x_k with true label y_k and each possible incorrect label $\bar{y}$. Computationally, only non-zero coefficients have to be represented. The gradient of Equation 7.2 with respect to $\mathbf{w}$ is given by

$$\nabla_{OP2} = 2\mathbf{w}\nabla_{\mathbf{w}} + C\sum_{i=1}^{n} \nabla_{\xi_i}.$$

Thus, applying Equation 7.5 gives us the first derivative in terms of the $\alpha_{ky_k\bar{y}}$

$$\frac{\partial OP2}{\partial \alpha_{ky_k\bar{y}}} = 2\mathbf{w}\frac{\partial \mathbf{w}}{\partial \alpha_{ky_k\bar{y}}} + C\sum_{i=1}^{n} \frac{\partial \xi_i}{\partial \alpha_{ky_k\bar{y}}}.$$

The partial derivative $\frac{\partial \mathbf{w}}{\partial \alpha_{ky_k\bar{y}}}$ resolves to $\Phi_{ky_k\bar{y}}$; that of ξ_i can be decomposed by the chain rule into

$$\frac{\partial \xi_i}{\partial \alpha_{ky_k\bar{y}}} = \frac{\partial \xi_i}{\partial \mathbf{w}}\frac{\partial \mathbf{w}}{\partial \alpha_{ky_k\bar{y}}} = \frac{\partial \xi_i}{\partial \mathbf{w}}\Phi_{ky_k\bar{y}} \ ,$$

$$\frac{\partial \xi_i}{\partial \mathbf{w}} = \sum_{\bar{y}\neq y_i} \frac{\partial \max\limits_{\tilde{y}\neq y_i} s(\tilde{y})}{\partial s(\bar{y})} \cdot \frac{\partial \ell_{\Delta(y_i,\bar{y})}\left(\langle \mathbf{w}, \Phi_{iy_i\bar{y}}\rangle\right)}{\partial \langle \mathbf{w}, \Phi_{iy_i\bar{y}}\rangle} \cdot \Phi_{iy_i\bar{y}}.$$

This solution generalizes binary unconstrained transductive learning (Chapelle, 2006) for general input-output spaces. The global minimum of Optimization Problem 2 can now easily be found with a standard gradient algorithm, such as conjugate gradient descent. By rephrasing the problem as an unconstrained optimization problem, its intrinsic complexity has not changed. We will observe the benefit of this approach in the following sections.

7.2 Unconstrained Transductive SVMs

In this section we present the transductive support vector machine (Vapnik, 1998) that has also been investigated by Bennet and Demiriz (1998) and Joachims (1999a). It consists of an EM-like self-training, wrapped around a support vector machine. In semi-supervised learning, unlabeled x_j^* for

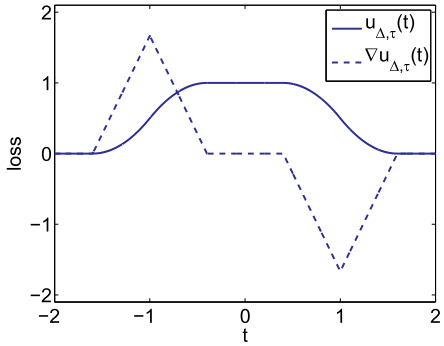


Figure 7.2: *Loss $u_{\Delta=1,\tau=0.6}(t)$ (solid) and first derivative (dashed).*

$n+1 \leq j \leq n+m$ are given in addition to the labeled pairs $(\mathbf{x}_i, y_i)$ for $1 \leq i \leq n$, where usually $n \ll m$. Optimization Problem 7.3 requires the unlabeled data to be classified by a large margin, but the actual label is unconstrained; this favors a low-density separation.

Optimization Problem 7.3 (TSVM) *Given a set of n labeled training pairs $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ and m unlabeled input examples $\mathbf{x}^*_{n+1}, \ldots, \mathbf{x}^*_{n+m}$, loss function Δ, and C, $C_u > 0$; the constrained TSVM optimization problem is defined as*

$$\min_{\mathbf{w},\boldsymbol{\xi}} \quad ||\mathbf{w}||^2 + C \sum_{i=1}^{n} \xi_i + C_u \sum_{j=n+1}^{n+m} \xi^*_j$$

subject to the constraints

$$\begin{aligned} \forall_{i=1}^{n} \forall_{\bar{y} \neq y_i} \langle \mathbf{w}, \Phi_{i y_i \bar{y}} \rangle &\geq \Delta(y_i, \bar{y}) - \xi_i \\ \forall_{j=n+1}^{n+m} \exists_{y^*_j} \forall_{\bar{y} \neq y^*_j} \langle \mathbf{w}, \Phi_{j y^*_j \bar{y}} \rangle &\geq \Delta(y^*_j, \bar{y}) - \xi^*_j \\ \forall_{i=1}^{n} \; \xi_i &\geq 0 \\ \forall_{j=n+1}^{n+m} \; \xi^*_j &\geq 0. \end{aligned}$$

Optimization problem 7.3 requires that there exists an y^*_j such that all other labels $\bar{y}$ violate the margin by no more than ξ^*_j. Hence, the value of slack variable ξ^*_j is determined by the label $\bar{y}$ that incurs the strongest margin violation. Alternatively, the sum of margin violations over all $\bar{y} \neq y^*_j$ may be upper bounded by ξ^*_j. In fact we can interpolate between max and sum by varying the softmax parameter ρ. Note that the optimum expansion $\boldsymbol{\alpha}$ is sparse, as only margin violating labels $\bar{y}$ contribute to the aggregation.

The constraints on ξ^*_j involve a *disjunction* over all possible labelings y^*_j of the unlabeled $\mathbf{x}^*_j$ which causes non-convexity and renders QP-solvers

not directly applicable. The TSVM implementation in SVM^{light} (Joachims, 1999a) treats the pseudo-labels y_j^* as additional discrete parameters. The existential quantifier is thus removed, but the criterion has to be minimized over all possible values of $(y_{n+1}^*, \ldots, y_{n+m}^*)$ and, in a nested step of convex optimization, over $\mathbf{w}$. Analogously to the ξ_i (Equation 7.1), we replace the constraints on ξ_j^*:

$$\begin{aligned}\xi_j^* &= \min_{y_j^*} \max \left\{ \max_{\bar{y} \neq y_j^*} \left\{ \Delta(y_j^*, \bar{y}) - \langle \mathbf{w}, \Phi_{j y_j^* \bar{y}} \rangle \right\}, 0 \right\} \\ &= \min_{y_j^*} \max_{\bar{y} \neq y_j^*} \left\{ u_{\Delta(y_j^*, \bar{y})}(\langle \mathbf{w}, \Phi_{j y_j^* \bar{y}} \rangle) \right\}. \end{aligned} \tag{7.6}$$

We quantify the loss induced by unlabeled instances $u_{\Delta,\tau}$ by a function slightly different from Huber loss $\ell_{\Delta,\epsilon}$. Diverging from ℓ, we engineer u to be symmetric, and to have a vanishing derivative at (and around) the point of symmetry. At this point, two labels score equally well (and better than all others), and the corresponding margin violation can be mitigated by moving $\mathbf{w}$ in two symmetric ways.

$$u_{\Delta,\tau}(t) = \begin{cases} 1 & : |t| \leq \Delta - \tau \\ 1 - \frac{(|t| - \Delta + \tau)^2}{2\tau^2} & : \Delta - \tau \leq |t| \leq \Delta \\ \frac{(|t| - \Delta - \tau)^2}{2\tau^2} & : \Delta \leq |t| \leq \Delta + \tau \\ 0 & : \text{otherwise} \end{cases}$$

$$\frac{\partial u_{\Delta,\tau}}{\partial t} = \begin{cases} 0 & : |t| \leq \Delta - \tau \\ -\frac{\operatorname{sgn}(t)}{\tau^2}(|t| - \Delta + \tau) & : \Delta - \tau \leq |t| \leq \Delta \\ +\frac{\operatorname{sgn}(t)}{\tau^2}(|t| - \Delta - \tau) & : \Delta \leq |t| \leq \Delta + \tau \\ 0 & : \text{otherwise.} \end{cases}$$

Having rephrased the constraints on ξ_j^* as an equation, we can pose the unconstrained transductive SVM optimization problem for structured outputs.

Optimization Problem 7.4 (∇TSVM) *Given a set of n labeled training pairs $(x_1, y_1), \ldots, (x_n, y_n)$ and m unlabeled input examples $x_{n+1}^*, \ldots, x_{n+m}^*$, loss function Δ, and $C, C_u > 0$; the unconstrained ∇TSVM optimization problem is defined as*

$$\min_{\mathbf{w}} \quad ||\mathbf{w}||^2 + C \sum_{i=1}^{n} \xi_i + C_u \sum_{j=n+1}^{n+m} \xi_j^* \tag{7.7}$$

where

$$\xi_i = \max_{\bar{y} \neq y_i} \ell_{\Delta(y_i,\bar{y})} \left(\langle \mathbf{w}, \Phi_{i y_i \bar{y}} \rangle \right)$$

$$\xi_j^* = \min_{y_j^*} \max_{\bar{y} \neq y_j^*} u_{\Delta(y_j^*,\bar{y})} \left(\langle \mathbf{w}, \Phi_{j y_j^* \bar{y}} \rangle \right).$$

Variables ξ_i and ξ_j^* remain in Optimization Problem 7.4 for notational harmony; they can be expanded to yield a closed, unconstrained optimization criterion.

Calculation of the ξ_j^* requires minimization over y_j^* and maximization over $\bar{y}$. For multi-class classification, this can be implemented by explicit enumeration. For general structured output spaces, y_j^* is determined by a decoder (e.g., a Viterbi algorithm for label sequence learning or a chart parser for parsing). For 0/1-loss, the worst margin violator $\bar{y}$ is the first runner-up, and a 2-best decoder can be used to find both y^* and $\bar{y}$ at the same time. For a Hamming-like loss, which shows the same Markov property as the feature map, $\bar{y}$ can be found by a modification of the decoder which rewards predicting wrong symbols.

Again we invoke the representer theorem 7.5 and optimize along the gradient $\frac{\partial OP4}{\partial \boldsymbol{\alpha}}$. In addition to the derivatives calculated in the Section 7.1, we need the partial derivatives of the ξ_j^*. They are analogous to those of ξ_i; let $\bar{s}(\tilde{y}) = u_{\Delta(y_j^*,\tilde{y})}(\langle \mathbf{w}, \Phi_{j y_j^* \tilde{y}} \rangle)$, we have

$$\frac{\partial \xi_j^*}{\partial \mathbf{w}} = \sum_{\bar{y} \neq y_j^*} \frac{\partial \operatorname{smax}_{\tilde{y} \neq y_j^*} \bar{s}(\tilde{y})}{\partial \bar{s}(\bar{y})} \cdot \frac{\partial u_{\Delta(y_j^*,\bar{y})} \left(\langle \mathbf{w}, \Phi_{j y_j^* \bar{y}} \rangle \right)}{\partial \langle \mathbf{w}, \Phi_{j y_j^* \bar{y}} \rangle} \cdot \Phi_{j y_j^* \bar{y}}.$$

Every expansion coefficient $\alpha^*_{j y_j^* \bar{y}}$ influences how strongly f favors label y_j^* over $\bar{y}$ for the unlabeled example j. This solution generalizes unconstrained transductive learning (Chapelle and Zien, 2005) for general input-output spaces.

Algorithmically, continuous optimization over all parameters $\alpha_{k y_k \bar{y}}$ is impossible due to exponentially many $\bar{y}$'s. However, our loss functions cause the solution to be sparse. In order to narrow the search to the non-zero variables, generalized ∇TSVM training interleaves two steps. In the decoding step, the algorithm iterates over all training instances and uses a 2-best decoder to produce the highest scoring output $\hat{y}$ and the worst margin violator $\bar{y} \neq \hat{y}$. For labeled examples (x_i, y_i), output $\hat{y}$ has to be equal to the desired y_i, and $\bar{y}$ must not violate the margin. Otherwise, the difference vector $\Phi_{i y_i \bar{y}}$ is added to the (initially empty) working set of the i-th example. For unlabeled data, the highest-scoring output of the joint classifier $\hat{y}_j^*$ serves as desired labeling and the runner-up as margin violator $\bar{y}_j$. Again, in the case of a margin violation, $\Phi_{j y^* \bar{y}}$ is added to the working set for x_j.

Table 7.1: *The ∇TSVM Algorithm*

Input: Labeled data $\{(x_i, y_i)\}_{i=1}^{n}$, unlabeled data $\{x_j^*\}_{j=n+1}^{n+m}$; parameters $C, C_u, \epsilon_\alpha > 0$.

repeat
  for each labeled example (x_i, y_i)
    $\bar{y} \leftarrow \text{argmax}_{y \neq y_i} \Delta(y_i, y) + \langle \mathbf{w}, \Phi(x_i, y) \rangle$
    if $\ell_{\Delta(y_i,\bar{y}),\epsilon}(\langle \mathbf{w}, \Phi(x_i, y_i) \rangle - \langle \mathbf{w}, \Phi(x_i, \bar{y}) \rangle) > 0$
      $W \leftarrow W \cup \{(i, y_i, \bar{y})\}$
    end
  end
  for each unlabeled example x_j^*
    $\hat{y}_j^* \leftarrow \text{argmax}_y \langle \mathbf{w}, \Phi(x_j^*, y) \rangle)$
    $\bar{y} \leftarrow \text{argmax}_{y \neq y_j^*} \Delta(y_j^*, y) + \langle \mathbf{w}, \Phi(x_j^*, y) \rangle$
    if $\exists y_j^* \in W \wedge y_j^* \neq \hat{y}_j^*$
      $\forall \bar{y} : W \leftarrow W \backslash \{(j, y_j^*, \bar{y})\}$
    end
    if $u_{\Delta(y_j^*,\bar{y}),\tau}(\langle \mathbf{w}, \Phi(x_j^*, y_j^*) \rangle - \langle \mathbf{w}, \Phi(x_j^*, \bar{y}) \rangle) > 0$
      $y_j^* \leftarrow \hat{y}_j^*$
      $W \leftarrow W \cup \{(j, y_j^*, \bar{y})\}$
    end
  end
  $\boldsymbol{\alpha} \leftarrow \text{argmin}_{\boldsymbol{\alpha}'} TSVM(\boldsymbol{\alpha}', W)$
  $\forall \alpha_{k y \bar{y}} < \epsilon_\alpha : W \leftarrow W \backslash \{(k, y_k, \bar{y})\}$
until convergence

Output: Optimized $\boldsymbol{\alpha}$, working set W.

In the optimization step, conjugate gradient descent (CG) is executed over the parameters $\alpha_{k y_k \bar{y}}$, given by all examples x_k, desired outputs y_k, and all associated pseudo-labels $\bar{y}$ currently in the working set. As proposed in Chapelle (2006) we use the kernel matrix as a preconditioner, which speeds up the convergence of the CG considerably. The inner loop of the ∇TSVM algorithm is depicted in Table 7.1. In an outer loop, ∇TSVM first increases C in a barrier fashion to avoid numerical instabilities, and eventually increases the the influence of the unlabeled examples C_u. The algorithm terminates when the working set remains unchanged over two consecutive iterations and C and C_u have reached the desired maximum value. Notice that ∇TSVM reduces to ∇SVM when no unlabeled examples are included in the training

process; i.e., for ∇SVM, lines 8-18 are removed from Table 7.1.

For binary TSVMs it has proven useful to add a *balancing constraint* to the optimization problem that ensures that the relative class sizes of the predictions are similar to those of the labeled points (Joachims, 1999a). For structured outputs, the relative frequencies of the output symbols $\sigma \in \Sigma$ may be constrained:

$$\frac{\sum_{j=n+1}^{n+m} \sum_{t=1}^{|x_j|} [[y_{j,t} = \sigma]]}{\sum_{j=n+1}^{n+m} |x_j|} = \frac{\sum_{i=1}^{n} \sum_{s=1}^{|x_i|} [[y_{i,s} = \sigma]]}{\sum_{i=1}^{n} |x_i|}.$$

Analoguously to binary TSVMs (Chapelle and Zien, 2005), this can be relaxed to "soft" linear constraints:

$$\sum_{j=n+1}^{n+m} \sum_{t=1}^{|x_j|} \left(\langle \mathbf{w}, \Phi(x_{j,t}, \sigma) \rangle + b_\sigma - \langle \mathbf{w}, \bar{\Phi}(x_{j,t}) \rangle + \bar{b} \right) = \hat{p}_\sigma$$

where $\Phi(x_{j,t}, \sigma)$ are the feature maps corresponding to predicting σ for position t of x_j, $\bar{\Phi}(x_{j,t}) = \sum_{\omega \in \Sigma} \Phi(x_{j,t}, \omega)/|\Sigma|$ is their average, the b_σ are newly introduced label biases with average $\bar{b} = \sum_\sigma b_\sigma/|\Sigma|$, and

$$\hat{p}_\sigma = (\sum_j |x_j|)(\sum_i \sum_s [[y_{is} = \sigma]]/(\sum_i |x_i|) - 1/|\Sigma|)$$

are centered predicted class sizes. By appropriately centering the unlabeled data, these constraints can be equivalently transformed into fixing the b_σ to constants. However, we do not implement any balancing here, as we empirically observe the fractions of predicted symbols to roughly agree with the corresponding fractions on the known labels.

7.3 Unconstraint CoSVM Optimization

The continuous optimization technique can also be applied to co-support vector machines. Recall that in the multi-view setting, two views Φ^1 and Φ^2 on the instances are available. The classifier combines two functions that operate in distinct views:

$$f(x, y) = f^1(x, y) + f^2(x, y),$$

with $f^v(x, y) = \langle \mathbf{w}^v, \Phi^v(x, y) \rangle$ for $v = 1, 2$. In the typical co-learning case, Φ^1 and Φ^2 are independent representations of x. By establishing a consensus between the hypotheses on the unlabeled data, co-learning minimizes an upper bound on the error rate.

The coSVM optimization problem requires that for each unlabeled instance $\mathcal{x}_j$ both classifiers agree on some label y by a large margin. The coSVM for structured output variables in Chapter 6 has *two* optimization criteria that are interleaved by discrete pseudo-labels for unlabeled examples that have to be exchanged iteratively between the views. Optimization Problem 7.5 integrates them into a joint criterion, again using existential quantification for the y_j. A parameter C_u quantifies the influence of the unlabeled data. We rephrase the constraints on ξ_i in Equation 7.1. For ξ_j^v, associated with unlabeled examples, we obtain a similar result,

$$\xi_j^v = \max\left\{\max_{\bar{y}\neq y_i^{\bar{v}}}\left\{\Delta(y_i^{\bar{v}},\bar{y}) - \langle \mathbf{w}^v, \Phi^v_{iy_i^{\bar{v}}\bar{y}}\rangle\right\}, 0\right\} = \max_{\bar{y}\neq y_i^{\bar{v}}} \ell_{\Delta(y_i^{\bar{v}},\bar{y})}\left(\langle \mathbf{w}^v, \Phi^v_{iy_i^{\bar{v}}\bar{y}}\rangle\right).$$

Again, the prediction $y_j^{\bar{v}}$ of the peer view $\bar{v}$ is treated as the true output for the j-th unlabeled input. Thus, we arrive at Optimization Problem 7.5.

Optimization Problem 7.5 (Unconstrained coSVM) *Given n labeled examples and m unlabeled examples, loss function Δ, joint feature mappings Φ^1 and Φ^2, $C, C_u > 0$; the unconstrained ∇coSVM optimization problem with margin-rescaling loss is defined as*

$$\min_{\mathbf{w}^1,\mathbf{w}^2} \quad \frac{1}{2}\sum_{v=1}^{2}||\mathbf{w}^v||^2 + C\sum_{v=1}^{2}\sum_{i=1}^{n}\xi_i^v + C_u\sum_{v=1}^{2}\sum_{j=n+1}^{n+m}\xi_j^v \tag{7.8}$$

where for $v = 1, 2$,

$$\begin{aligned}
\xi_i^v &= \max_{\bar{y}\neq y_i} \ell_{\Delta(y_i,\bar{y})}(\langle \mathbf{w}^v, \Phi^v_{iy_i\bar{y}}\rangle)\\
\xi_j^v &= \max_{\bar{y}\neq \hat{y}_j^{\bar{v}}} \ell_{\Delta(\hat{y}_j^{\bar{v}},\bar{y})}(\langle \mathbf{w}^v, \Phi^v_{j\hat{y}_j^{\bar{v}}\bar{y}}\rangle)\\
\hat{y}_j^v &= \operatorname*{argmax}_{\bar{y}\in\mathcal{Y}(\mathcal{x}_j)}\langle \mathbf{w}^v, \Phi^v(\mathcal{x}_j,\bar{y})\rangle.
\end{aligned}$$

Optimization Problem 7.5 has a single, closed-form optimization criterion. Replacing the Huber loss for the hinge loss and the maximum by the softmax makes the criterion differentiable, and optimization can follow the gradient $\frac{\partial OP5}{\partial \boldsymbol{\alpha}}$. In addition to the derivatives calculated in the Section 7.1, we need the partial derivatives of the ξ_j^v. They are analogous to those of ξ_i for each $v = 1, 2$. We have

$$\frac{\partial \xi_j^v}{\partial \mathbf{w}^v} = \sum_{\bar{y}\neq y_j^{\bar{v}}} \frac{\partial \operatorname{smax}_{\tilde{y}\neq y_j^{\bar{v}}} s(\tilde{y})}{\partial s(\bar{y})} \ell'_{\Delta(y_j^{\bar{v}},\bar{y})}\left(\langle \mathbf{w}^v, \Phi^v_{jy_j^{\bar{v}}\bar{y}}\rangle\right)\Phi^v_{jy_j^{\bar{v}}\bar{y}}.$$

Every expansion coefficient $\alpha^v_{j y_j^{\bar{v}} \bar{y}}$ influences how strongly f favors the peer labeling $y_j^{\bar{v}}$ over $\bar{y}$ for the unlabeled example j.

As ∇TSVM, unconstrained coSVM training interleaves two steps. In the decoding step, the algorithm iterates over all training instances and uses a 2-best decoder to produce the highest-scoring output $\hat{y}$ and the runner-up $\bar{y} \neq \hat{y}$. For labeled examples $(\mathcal{x}_i, y_i)$, output $\hat{y}$ has to be equal to the desired y_i and $\bar{y}$ must not violate the margin. Otherwise, the highest-scoring margin violator is added to the (initially empty) working set of negative pseudo-labels for the i-th example. For unlabeled data, the highest-scoring output of the joint classifier $\hat{y}_j$ serves as desired label. If $\bar{y}_j^v$ violates the margin in any view, both $\hat{y}_j$ and $\bar{y}_j^v$ are added to the working set for the j-th example.

In the optimization step, conjugate gradient descent is executed over the parameters $\alpha_{k y_k \bar{y}}$, given by all examples $\mathcal{x}_k$, desired outputs y_k, and all negative pseudo-labels $\bar{y}$ currently in the working set of the k-th example. After each optmization step, the influence of the unlabeled examples C_u is increased as well as the parameter of the barrier approach to guarantee a fast convergence. The algorithm terminates when no difference vectors are added to any working set in an iteration. Table 7.2 depicts the inner loop of ∇coSVM.

7.4 Experiments

We investigate unconstrained optimization of structured output support vector machines by comparing differentiable ∇SVM and ∇TSVM to SVMs solved by constrained, quadratic programming (QP) approaches. In each setting, the influence of unlabeled examples is determined by a smoothing strategy which exponentially approaches C_u after a fixed number of epochs. We optimize C_u using resampling and then fix C_u and present curves that show the average error over 100 randomly drawn training and holdout sets; error bars indicate standard error. In all experiments we set $C = 1$, $\epsilon = 0.3$, and $\tau = 0.4$.

7.4.1 Execution Time

Figure 7.3 compares the execution times of CG-based ∇SVM and ∇TSVM to a QP-based SVM where we use the same convergence criteria for all optimizers. ∇TVSM is trained with the respective number of labeled examples and a five-fold larger set of unlabeled instances. In addition to being faster than a solution based on solving QPs, the continuous optimization is remarkably efficient at utilizing the unlabeled data. For instance, ∇TSVM with 50 labeled and 250 unlabeled examples converges considerably faster than ∇SVM

Table 7.2: *The ∇coSVM Algorithm*

Input: Labeled data $\{(x_i, y_i)\}_{i=1}^{n}$, unlabeled data $\{x_j\}_{j=n+1}^{n+m}$; parameters C, C_u, $r_{max} > 0$.

repeat
  for each labeled example (x_i, y_i) **and** $v = 1, 2$
    $\bar{y}^v \leftarrow \text{argmax}_{y \neq y_i} \Delta(y_i, y) + \langle \mathbf{w}^v, \Phi^v(x_i, y) \rangle$
    if $\ell_{\Delta(y_i, \bar{y}), \epsilon}(\langle \mathbf{w}^v, \Phi^v(x_i, y_i) \rangle - \langle \mathbf{w}^v, \Phi^v(x_i, \bar{y}^v) \rangle) > 0$
      $W^v \leftarrow W^v \cup \{(i, y_i, \bar{y}^v)\}$
    end
  end
  for each unlabeled example x_j **and** $v = 1, 2$
    $\hat{y}_j^{\bar{v}} \leftarrow \text{argmax}_y \langle \mathbf{w}^{\bar{v}}, \Phi^{\bar{v}}(x_j, y) \rangle$
    $\bar{y}^v \leftarrow \text{argmax}_{y \neq y_j^{\bar{v}}} \Delta(y_j^{\bar{v}}, y) + \langle \mathbf{w}^v, \Phi^v(x_j, y) \rangle$
    if $\exists y_j^{\bar{v}} \in W^v \wedge y_j^{\bar{v}} \neq \hat{y}_j^{\bar{v}}$
      $\forall \bar{y} : W^v \leftarrow W^v \backslash \{(j, y_j^{\bar{v}}, \bar{y})\}\%\backslash\text{hspace } \{7.8cm\}$
    end
    if $\ell_{\Delta(\hat{y}_j^{\bar{v}}, \bar{y}^v)}(\langle \mathbf{w}^v, \Phi^v(x_j, y_j^{\bar{v}}) \rangle - \langle \mathbf{w}^v, \Phi^v(x_j, \bar{y}^v) \rangle) > 0$
      $y_j^{\bar{v}} \leftarrow \hat{y}_j^{\bar{v}}$
      $W^v \leftarrow W^v \cup \{(j, y_j^{\bar{v}}, \bar{y}^v)\}$
    end
  end
  $\boldsymbol{\alpha} \leftarrow \text{argmin}_{\boldsymbol{\alpha}'} \, coSVM(\boldsymbol{\alpha}', W^1, W^2)$
until convergence **or** r_{max} iterations

Output: Optimized $\boldsymbol{\alpha}$ working sets W^1 and W^2.

and qpSVM with only 200 labeled instances.

7.4.2 Multi-class Classification

For the multi-class classification experiments, we use a cleaned variant of the Cora data set that contains 9,555 linked computer science papers with a reference section. The data set is divided into eight different classes. We extract term frequencies of the document and of the anchor text of the inbound links. The latter are drawn from three sentences, centered at the occurrence of the reference. We compare the performances of ∇TSVM with 0/1 loss to the performance of TSVM^{light} (Joachims, 1999b), trained with a one-against-all strategy. Figure 7.4 details the error-rates for 200 labeled examples and varying numbers of unlabeled instances. With no unlabeled

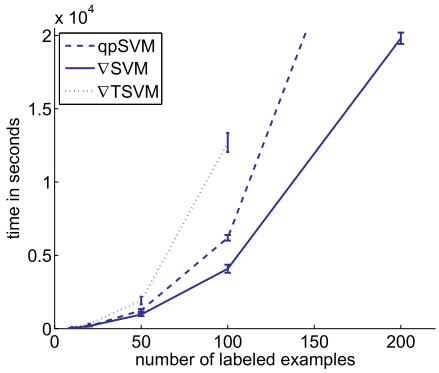


Figure 7.3: *Execution time.*

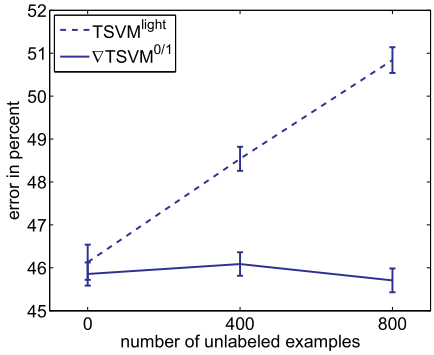


Figure 7.4: *Error rates for the Cora data set.*

data, both transductive methods reduce to their fully-supervised, inductive counterparts. Both SVMs perform equally well for the labeled instances. However, when unlabeled examples are included into the training process, the performance of TSVM^{light} deteriorates. The error-rates of ∇TSVM show a slight improvement with 800 unlabeled instances.

We also apply our method to the six-class dataset COIL as used in Chapelle et al. (2006a), and compare to the reported one-against-all TSVM results. For $n = 10$ labeled points, we achieve 68.87% error, while the one-against-all TSVM achieves 67.50%. For $n = 100$ points, the results are 25.42% as compared to 25.80%.

7.4.3 Artificial Sequential Data

The artificial galaxy data set (Lafferty et al., 2004) consists of 100 sequences of a length of 20, generated by a two state hidden Markov model. The initial

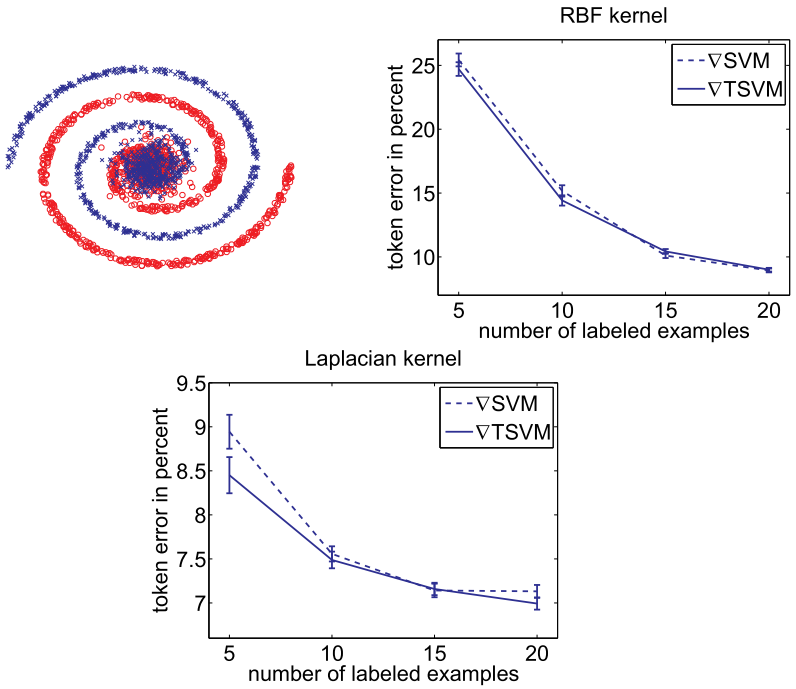


Figure 7.5: *The galaxy data set (top left) and error rates for ∇SVM and ∇TSVM using RBF (top right) and graph kernels (bottom).*

state is chosen uniformly and there is a 10% chance of switching the state. Each state emits instances uniformly from one of the two classes (see Figure 7.5; top left).

We run ∇SVM and ∇TSVM using Hamming loss with two different kernels, a Gaussian RBF kernel with bandwidth $\sigma = 0.35$ and a semi-supervised graph kernel. The graph kernel is constructed from a 10-nearest neighbor graph and given by $K = 10\,(L + \mathbb{1}\rho)^{-1}$, with graph Laplacian L and $\rho = 10^{-6}$ as proposed by Lafferty et al. (2004).

In each experiment we draw a certain number of labeled sequences at random and use the rest either as unlabeled examples or as holdout set. We report the averages over 20 runs. Figure 7.5 (top right and bottom) details the results for the semi-supervised vs. supervised algorithm and the semi-supervised vs. standard kernel. Since the approaches are orthogonal, we apply all four combinations. For increasing numbers of labeled examples, the error rates of the tested models decrease. The continuous TSVM performs slightly better than the supervised SVM; the differences are significant in only a few cases. This problem is very well tailored for the Laplacian kernel. The error rates achieved with the semi-supervised kernel are between 3% to 20% lower than the corresponding results for the RBF kernel.

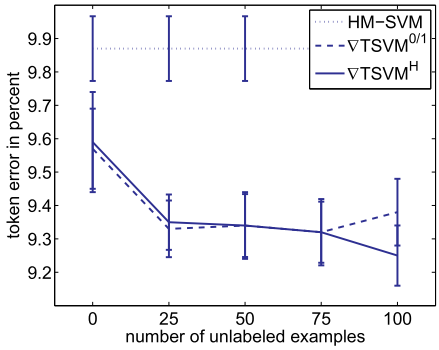


Figure 7.6: *Token error for the Spanish news wire data set with 10 labeled instances.*

7.4.4 Named Entity Recognition

The CoNLL2002 data consists of sentences from a Spanish news wire archive and contains nine label types which distinguish between person, organization, location, and other names. We use $3,100$ sentences of between 10 and 40 tokens, leading to $\approx 24,000$ distinct tokens in the dictionary. Moreover, we extract surface clue features, such as capitalization and others. We use a window of size three, centered around each token.

In each experiment we randomly draw a specified number of labeled and unlabeled training and holdout data without replacement in each iteration. We ensure that each label occurs at least once in the labeled training data, otherwise, we discard and draw again. We compare ∇TSVM with 0/1 loss and Hamming loss to the HM-SVM (Altun et al., 2003b), trained by incrementally solving quadratic programs over subspaces associated with individual input examples. Figure 7.6 details the results for 10 labeled sequences.

∇SVM converges to better local optima than HM-SVM. We credit this finding to global conjugate gradient based optimization compared to solving local quadratic programs. When unlabeled examples are included in the training process, the error of the ∇TSVM decreases significantly. ∇TSVM^H with Hamming loss performs slightly better than $\nabla\text{TSVM}^{0/1}$ using 0/1 loss.

7.5 Discussion

The TSVM criterion is non-convex and the minimization can be difficult even for binary class variables. In order to scale the TSVM to structured outputs, we employ a technique that eliminates the discrete parameters and allows for a conjugate gradient descent in the space of expansion coefficients $\boldsymbol{\alpha}$. Empir-

ical comparisons of execution time show that the continuous approaches are more efficient than standard approaches based on quadratic programming.

For the Cora text classification problem, transductive learning does not achieve a substantial benefit over supervised learning. Worse yet, the combinatorial TSVM increases the error substantially, whereas ∇TSVM has a negligible effect. In order to provide an unbiased account, we present this finding with emphasis equal to any positive result. For the Spanish news named entity recognition problem, we consistently observe small but significant improvements over purely supervised learning.

One might expect transductive learning to outperform supervised learning because more information is available. However, these test instances introduce non-convexity, and the local minimum retrieved by the optimizer may be worse than the global minimum of the convex supervised problem. Our experiments indicate that this might occasionally occur.

For the galaxy problem, the benefit of ∇TSVM over ∇SVM is marginal, and observable only for very few labeled examples. By its design this problem is very well suited for graph kernels, which reduce the error rate by 50%. In the graph Laplacian approach (Sindhwani et al., 2005b), an SVM is trained on the labeled data, but in addition to the standard kernel, the graph Laplacian derived from labeled and unlabeled points serves as a regularizer. For binary classification, combining TSVM and graph Laplacian yields the greatest benefits (Chapelle and Zien, 2005). For structured variables, we observe a similar effect, though much weaker.

The presented ∇TSVM rests on a cluster assumption for entire structures, while graph-based methods (Lafferty et al., 2004; Altun et al., 2006) exploit the distribution of parts of structures. Both approaches improve over supervised learning on some datasets and fail to do so on others. This raises the question of how to determine which kind of assumptions are appropriate for a given task.

7.6 Comparison with CoSVMs

Figure 7.7 shows a comparison of the performance of ∇TSVMs and coSVMs for multi-class classification named entity recognition tasks, respectively. We utilized the Cora and the Spanish news wire data sets. In the latter we refrain from employing a window of size 3 around each token for computational reasons. The respective experimental settings are detailed in Section 6.5.1 (multi-class) and 6.5.2 (NER).

Figure 7.7 (left) shows the results for the Cora multi-class classification task. Both semi-supervised algorithms hardly make use of the inclusion of

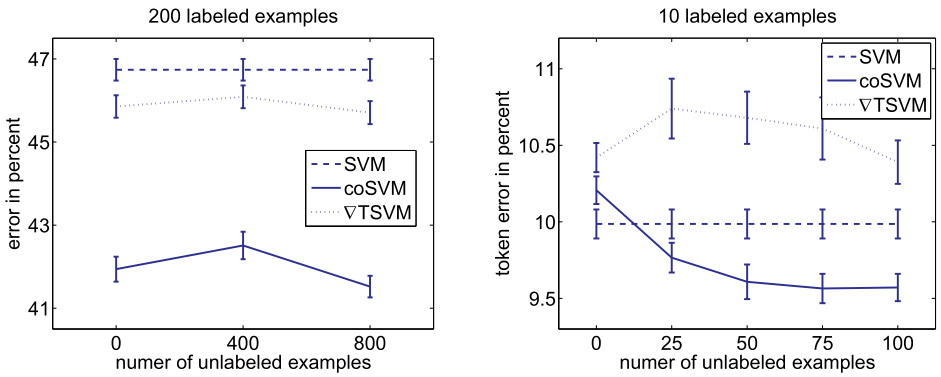


Figure 7.7: *Comparison of ∇TSVM and coSVM. Left: Results for the Cora data set. Right: Results for the Spanish news wire data set.*

unlabeled examples. However, coSVM leads to significantly lower error rates than ∇TSVM. The difference in performance is about 4%.

The right hand side of Figure 7.7 details the results for the Spanish news wire named entity recognition task. CoSVM effectively incorporates unlabeled examples and we observe a negative correlation of the number of unlabeled examples and the token error; that is, if we add unlabeled examples, the error decreases further. By contrast, the transductive SVM cannot make use of the inclusion of unlabeled data. The performance drops for only a few unclassified sequences and recovers slightly when more unlabeled sequences are included.

7.7 Conclusions

We devised a transductive support vector machine for structured variables (∇TSVM). We transformed the original combinatorial and constrained optimization problem into a differentiable and unconstrained one. The resulting optimization problem is still non-convex but can be optimized efficiently, for instance via a conjugate gradient descent. A differentiable variant of the SVM for structured variables (∇SVM) is obtained for the special case of a fully labeled training set.

We applied both methods with various loss functions to multi-class classification and sequence labeling problems. Based on our empirical findings, we can rule out the hypothesis that ∇TSVM generally improves learning with structured output variables over purely supervised learning, as well as the hypothesis that ∇TSVM never improves accuracy. Moreover, a direct comparison with coSVMs indicated that including unlabeled examples ac-

cording to the consensus maximization principle outclasses the principle of transduction in many settings.

Chapter 8

Supervised Clustering of Streaming Data for Email Batch Detection

This chapter deals with a case study on email batch detection. Although batch detection this is actually an unsupervised task, a ground truth of correct clusterings exists and essentially we arrive at an instance of semi-supervised learning.

Senders of spam, phishing, and virus emails avoid mailing multiple identical copies of their messages. Once a message is known to be malicious, all subsequent identical copies of the message could be blocked easily, and without any risk of erroneously blocking regular emails. Collective features of jointly generated batches of messages could provide additional hints for automatic classification, if batches could be recognized as such. Tools for spam, phishing, and virus dissemination employ templates and stochastic grammars, for text messages as well as for images and the source code of viruses. The templates are instantiated for each message. Table 8.1 shows two illustrative spam messages, generated from the same template.

A natural approach to identifying batches in incoming messages is to cluster groups of similar instances. But unlike for exploratory data analysis, a *ground truth* of *correct clusterings* exists. In order to decide which technique to use, one has to consider the characteristics of electronic messaging. The overall amount of spam in electronic messages is estimated to be approximately 80 percent. Currently, 80 to 90 percent of these messages are generated by only a few spam senders, each of them maintaining a small number of templates at a time, but exchanging them rapidly. Thus, examining the total email traffic of a short time window, the bulk of incoming messages has been generated by a small number of templates while the remaining 20

Table 8.1: *Two spam mails from the same batch*

Hello, This is Terry Hagan.We are accepting your mo rtgage application. Our company confirms you are legible for a $250.000 loan for a $380.00/month. Approval process will take 1 minute, so please fill out the form on our website: http://www.competentagent.com/application/ Best Regards, Terry Hagan; Senior Account Director Trades/Fin ance Department North Office
Dear Mr/Mrs, This is Brenda Dunn.We are accepting your mortga ge application. Our office confirms you can get a $228.000 lo an for a $371.00 per month payment. Follow the link to our website and submit your contact information. Easy as 1,2,3. http://www.competentagent.com/application/ Best Regards, Brenda Dunn; Accounts Manager Trades/Fin ance Department East Office

percent cover newsletters, personal, and business communications. In a clustering solution, the latter would result in a large number of singleton clusters while newsletters and spam batches would lead to many large groups. An appropriate clustering algorithm needs to allow for an arbitrary number of clusters and an adjustable similarity measure that can be adapted to yield the ground truth of correct clusterings.

Initially, correlation clustering meets all these requirements. Finley and Joachims (2005) adapt the similarity measure of correlation clustering with structural support vector machines. The solution is equivalent to a poly-cut in a fully connected graph spanned by the messages and their pairwise similarities. However, this solution ignores the temporal structure of the data. And although training can be performed offline, the correlation clustering procedure has to make a decision for each incoming message in real time as to whether or not it is part of a batch. Larger email service providers have to deal with an amount of emails in the order of 10^8 emails each day. Being cubic in the number of instances, this solution leads to intractable problems in practice. We devise a sequential clustering technique that overcomes these

drawbacks. Exploiting the temporal nature of the data, it is linear in the number of instances. Sequential clustering can easily be integrated in structural SVMs, allowing for the similarity measure to be adapted for a labeled training set.

This chapter is structured as follows. We discuss related work in Section 8.1 and introduce the problem setting in Section 8.2 where we also we derive a learning method starting from a relaxed clustering variant. In Section 8.3, we exploit the temporal nature of the data and devise a sequential clustering algorithm with an appropriate learning variant. We report on experimental results in Section 8.4. Section 8.5 provides a conclusion.

8.1 Related Work

Prior work on clustering of streaming data mainly focused on finding single-pass approximations to k-Center algorithms. Guha et al. (2003) develop a constant-factor approximation for k-Median clustering, whereas Ordonez (2003) use an incremental version of k-Means for clustering streams of binary data.

Prior information about the clustering structure of a data set allows for enhancements to clustering algorithms such as k-Means. For instance, Wagstaff et al. (2001) incorporate the background knowledge as must-link and cannot-link constraints into the clustering process. Bar-Hillel et al. (2003) and Xing et al. (2002) demonstrate the learning of a metric over the data space that incorporates prior knowledge. Using batch information for spam classification has been studied for settings where multiple users receive spam emails from the same batch. Gray and Haahr (2004) as well as Damiani et al. (2004) discuss the difficulties concerning the distribution of batch information and trust between users, while mostly heuristics are used to identify duplicate emails from the same batch.

More sophisticated exploration of robust identification of duplicates has been done in other domains, for instance in terms of fixed similarity measures, such as the fraction of matching words (Cooper et al., 2002a,b) and sentences (Brin et al., 1995). Other applications include the identification of duplicates in data bases (Bilenko and Mooney, 2003), and in centralized (Kolcz et al., 2004) and decentralized networks (Zhou et al., 2003). Learning adaptive similarity measures from data has previously been studied by Ristad and Yianilos (1997). Correlation clustering with fully connected graphs is introduced in (Bansal et al., 2002). A generalization to arbitrary graphs is presented in Charikar et al. (2005) and Emanuel and Fiat (2003) show the equivalence to a poly-cut problem. Approximation strategies for the

NP-complete decoding are presented in Demaine and Immorlica (2003) and Swamy (2004). Finley and Joachims (2005) investigated supervised clustering with structural support vector machines and Joachims and Hopcroft (2005) examined upper bounds on the error of correlation clustering.

Several discriminative algorithms have been studied that utilize joint spaces of input and output variables. These include max-margin Markov models (Taskar et al., 2004a), kernel conditional random fields (Lafferty et al., 2004), and support vector machines for structured and interdependent output spaces (Tsochantaridis et al., 2005). These methods utilize kernels to compute the inner product in input-output space. This approach allows for the capturing of arbitrary dependencies between inputs and outputs. An application-specific learning method is constructed by defining appropriate features, and choosing a decoding procedure that efficiently calculates the argmax, exploiting the dependency structure of the features.

8.2 Learning to Cluster

The goal is to find a procedure that assigns a cluster membership to each new email of a stream. A ground truth exists for this clustering problem: All jointly created instances of the same template should be grouped. Individually written emails should form singleton clusters. The ground truth is observable for the training set.

Thus, we are given n *sets* of training instances $\mathbf{x}_1, \ldots, \mathbf{x}_n$, $\mathbf{x}_i \in \mathcal{X}$, where the i-th set $\mathbf{x}_i$ consists of T_i messages $\mathbf{x}_i = \{x_{i1}, \ldots, x_{iT_i}\}$ with $x_{ij} \in \Omega$. For each set we are also given the correct clustering as an adjacency matrix $\mathbf{y}_i \in \mathcal{Y}(\mathbf{x}_i)$, with $[\mathbf{y}_i]_{jk} = 1$ if messages x_{ij} and x_{ik} are elements of the same cluster, and 0 otherwise. The output alphabet is binary, $\Sigma = \{0, 1\}$. The joint feature representation is detailed in Section 3.2.4 and defined as

$$\Phi(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{T} \sum_{k=1}^{j-1} [\mathbf{y}]_{jk} \psi(x_j, x_k), \tag{8.1}$$

where $\psi : \Omega \times \Omega \to \mathbb{R}^d$ denotes the vector of pairwise feature functions drawn from pairs of objects. Examples of components of ψ are the *tf.idf similarity of the message bodies, the edit distance of the subject lines, or the similarity of color histograms of images included in the messages.*

As mentioned in Section 3.2.4, the decoding problem in Equation 8.2 (see also Equations 3.36-3.38) of finding the adjacency matrix which maximizes

the inner-cluster similarities is NP-complete,

$$\hat{y} = \operatorname*{argmax}_{\bar{y}\in\mathcal{Y}(x_i)} f(x_i, \bar{y}) = \operatorname*{argmax}_{\bar{y}\in\mathcal{Y}(x_i)} \langle \mathbf{w}, \Phi(x_i, \bar{y})\rangle. \tag{8.2}$$

A common approach is to approximate the discrete variables by relaxing the binary edge labels $[y]_{jk}$ *to continuous variables* $[z]_{jk} \in [0,1]$*. We obtain the optimization problem*

$$\begin{aligned} \max_{\bar{z}\in\mathcal{Z}(x)} \quad & \langle \mathbf{w}, \Phi(x, \bar{z})\rangle \\ s.t. \quad & \forall_{jkl}\, (1 - [\bar{z}]_{jk}) + (1 - [\bar{z}]_{kl}) \geq (1 - [\bar{z}]_{jl}) \\ & \forall_{jk}\, [\bar{z}]_{jk} \in [0,1]. \end{aligned}$$

We refer to this decoding strategy as LP decoding. Taking a margin-rescaling approach and substituting the normalized loss function Δ_{total} into the approximate inference problem leads to the loss augmented approximate inference problem (Taskar et al., 2005)

$$\begin{aligned} & \max_{\bar{z}\in\mathcal{Z}(x)} \Delta_{total}(y, \bar{z}) + \langle \mathbf{w}, \Phi(x, \bar{z})\rangle \\ &= \max_{\bar{z}\in\mathcal{Z}(x)} \sum_{k<j} \frac{|[y]_{jk} - [\bar{z}]_{jk}|}{\sum_{k'\neq j}[y]_{k'k}} + \langle \mathbf{w}, \Phi(x, \bar{z})\rangle \\ &= \max_{\bar{z}\in\mathcal{Z}(x)} \underbrace{\sum_{j,k<j} \frac{[y]_{jk}}{\sum_{k'\neq j}[y]_{k'k}}}_{=:\kappa} + \sum_{j,k<j} [\bar{z}]_{jk} \Big(\langle \mathbf{w}, \boldsymbol{\psi}(x_j, x_k)\rangle - \underbrace{\frac{2[y]_{jk} - 1}{\sum_{k'\neq j}[y]_{k'k}}}_{=:[\boldsymbol{\zeta}]_{jk}} \Big), \end{aligned}$$

where $\bar{z}$ ranges over all relaxed adjacency matrices which satisfy

$$\begin{aligned} \forall j,k,l: \ [\bar{z}]_{jk} + [\bar{z}]_{kl} - [\bar{z}]_{jl} - 1 \leq 0 \\ \forall i,k: \ [\bar{z}]_{jk} - 1 \leq 0 \\ \forall i,k: \ -[\bar{z}]_{jk} \leq 0. \end{aligned}$$

Integrating these constraints into the objective function leads to the corresponding Lagrangian

$$\begin{aligned} L(\bar{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}) =& \kappa + \sum_{j,k<j} [\bar{z}]_{jk} \left(\langle \mathbf{w}, \psi(x_j, x_k)\rangle - [\boldsymbol{\zeta}]_{jk} \right) \\ & - \sum_{j,k,l} \lambda_{jkl} ([\bar{z}]_{jk} + [\bar{z}]_{kl} - [\bar{z}]_{jl} - 1) \\ & - \sum_{j,k} \mu_{jk} ([\bar{z}]_{jk} - 1) + \sum_{j,k} \nu_{jk} [\bar{z}]_{jk} \\ =& \kappa + \boldsymbol{\lambda}^\mathsf{T}\mathbf{1} + \boldsymbol{\mu}^\mathsf{T}\mathbf{1} + (\mathbf{S}\mathbf{w} - \boldsymbol{\zeta} - \mathbf{A}\boldsymbol{\lambda} - \boldsymbol{\mu} + \boldsymbol{\nu})^\mathsf{T} \bar{z}, \end{aligned}$$

where we used $\mathbf{S}$ shorthand for the pairwise similarity matrix with elements $[\mathbf{S}]_{jk,m} = \psi_m(x_j, x_k)^\mathsf{T}$ and $\mathbf{A}$ is a coefficient matrix defined as

$$[\mathbf{A}]_{j'k',jkl} = \begin{cases} +1 & : \text{ if } (j' = j \wedge k' = k) \vee (j' = k \wedge k' = l) \\ -1 & : \text{ if } j' = j \wedge k' = l \\ 0 & : \text{ otherwise.} \end{cases}$$

The substitution of the derivatives with respect to the components of $\boldsymbol{z}$ into the Lagrangian and elimination of $\boldsymbol{\nu}$ removes its dependence on the primal variables and we resolve the corresponding dual that is given by

$$\begin{aligned} \min_{\boldsymbol{\lambda},\boldsymbol{\mu}} \quad & \kappa + \boldsymbol{\lambda}^\mathsf{T}\mathbf{1} + \boldsymbol{\mu}^\mathsf{T}\mathbf{1} \\ \text{s.t.} \quad & (\mathbf{S}\mathbf{w} - \boldsymbol{\zeta} - \mathbf{A}^\mathsf{T}\boldsymbol{\lambda} - \boldsymbol{\mu}) \leq \mathbf{0} \\ & \boldsymbol{\lambda}_{jkl}, \boldsymbol{\mu}_{jk} \geq \mathbf{0}. \end{aligned}$$

Strong duality holds and the minimization over $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ can be combined with the minimization over $\mathbf{w}$. The integration into structured support vector machines finally leads to Optimization Problem 8.1.

Optimization Problem 8.1 (LP-SVM) *Given a set of n labeled clusterings $(\mathcal{X}_1, y_1), \ldots, (\mathcal{X}_n, y_n)$ and $C > 0$, the integrated LP approximation SVM optimization problem is defined as*

$$\min_{\mathbf{w},\boldsymbol{\xi},\boldsymbol{\lambda}_i,\boldsymbol{\mu}_i} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

subject to the constraints

$$\begin{aligned} \forall_{i=1}^{n} \ \langle \mathbf{w}, \Phi(\mathcal{X}_i, y_i)\rangle + \xi_i &\geq \kappa_i + \boldsymbol{\lambda}_i^T\mathbf{1} + \boldsymbol{\mu}_i^T\mathbf{1} \\ \forall_{i=1}^{n} \ \mathbf{S}_i\mathbf{w} - \boldsymbol{\zeta}_i &\leq \mathbf{A}_i^T\boldsymbol{\lambda}_i + \boldsymbol{\mu}_i \\ \forall_{i=1}^{n}\forall_{jkl=1}^{T_i} \ [\boldsymbol{\lambda}_i]_{jkl} &\geq 0 \\ \forall_{i=1}^{n}\forall_{jk=1}^{T_i} \ [\boldsymbol{\mu}_i]_{jk} &\geq 0. \end{aligned}$$

Optimization Problem 8.1 can be solved directly using standard QP-solvers. Because of the cubic number of triangle inequalities, the number of Lagrange multipliers $[\boldsymbol{\lambda}_i]_{jkl}$ in Optimization Problem 8.1 is cubic in the number of emails T_i per set. Finley and Joachims (2005) chose a similar approach but arrive at an iterative algorithm to learn the weight vector. The iterative algorithm represents only a subset of the constraints and therefore achieves a speed-up at training time. In our case, the training samples are modestly sized whereas, at application time, a high-speed stream has to be processed. Therefore, we will develop a linear decoder in the next section. The linear decoder will also reduce the complexity of the parameter optimization problem from cubic to quadratic.

Table 8.2: *Sequential Clustering Algorithm*

Input: Messages $x_1, \ldots, x_T$

```
Initialize C ← {}
for j = 1,...,T
    c_j ← argmax_{c∈C} Σ_{x_k∈c} ⟨w, Φ(x_k, x_j)⟩
    if Σ_{x_k∈c_j} ⟨w, Φ(x_k, x_j)⟩ < 0
        C ← C ∪ {{x_j}}
    else
        C ← C \ {c_j} ∪ {c_j ∪ {x_j}}
    endif
end
```

Output: Clustering $\mathcal{C}$.

8.3 Clustering of Streaming Data

In our batch detection application, incoming emails are processed sequentially. The decision on the cluster assignment has to be made immediately, within an SMTP session, and cannot be altered thereafter. Because of the high volume of the email stream, any decoding algorithm requiring more than linear execution time in the number of emails processed would be prohibitive.

We therefore impose the constraint that cluster membership cannot be reconsidered once a decision has been made in the decoding procedure. When the partitioning of all previous emails in the window is fixed, a new message is processed by either assigning it to one of the existing clusters, or creating a new singleton batch. Algorithm 8.2 details this approach; the initially empty partitioning $\mathcal{C}$ becomes a singelton cluster when the first message arrives. Every new message then either groups onto an existing cluster c_j or extends $\mathcal{C}$ by forming its own singelton cluster $\{x_t\}$, respectively.

In general, given a fixed clustering of $x_1, \ldots, x_{T-1}$, the decoding problem of finding the y that maximizes Equation 3.35 reduces to

$$\max_{y} \sum_{j=1}^{T} \sum_{k=1}^{j-1} [y]_{jk} sim(x_j, x_k) = \max_{y} \sum_{j=1}^{T-1} \sum_{k=1}^{j-1} [y]_{jk} sim(x_j, x_k) + [y]_{Tk} sim(x_T, x_k). \tag{8.3}$$

The first term depends only on the clustering of previous messages and is constant. Finding the maximum in Equation 8.3 therefore amounts to as-

signing it to the cluster which is most similar to x_T or, if no existing cluster has positive total similarity, establishing a new singleton cluster.

In terms of the adjacency matrix y_i of the i-th input, the task is to find entries for the T-th row and column, realizing the optimal clustering of x_T. We denote the set of matrices that are consistent clusterings and are equal to the i-th example, y_i, in all rows/columns except for the T-th row/column, by $\mathcal{Y}_T(x_i)$. If we denote the potential new cluster (which is empty before inserting x_T) with $\bar{c}$, $\mathcal{Y}_T(x_i)$ is of the size $|\mathcal{C} \cup \{\bar{c}\}| \leq T_i$. Finding the new optimal clustering can be expressed as the following maximization problem.

Decoding Strategy 8.1 *Given T_i instances $x_1, \ldots, x_{T_i}$, similarity measure $\text{sim}_{\mathbf{w}} : (x_j, x_k) \mapsto r \in \mathbb{R}$, and a clustering of instances $x_1, \ldots, x_{T_i-1}$, the sequential decoding problem is defined as*

$$\hat{y} = \operatorname*{argmax}_{\bar{y} \in \mathcal{Y}_T(x_i)} \sum_{k=1}^{T_i-1} [\bar{y}]_{T_i k} \text{sim}_{\mathbf{w}}(x_{T_i}, x_k). \tag{8.4}$$

Now we derive an optimization problem that requires the sequential clustering to produce the correct output for all training data. Optimization Problem 8.2 constitutes a compact formulation for finding the desired optimal weight vector by treating every message as the most recent message, in order to utilize the available training data as effectively as possible.

Optimization Problem 8.2 *Given n labeled clusterings, $C > 0$, the sequential SVM optimization problem is defined as*

$$\min_{\mathbf{w}, \boldsymbol{\xi}} \quad \frac{1}{2}\|\mathbf{w}\|^2 \; + \; C \sum_{i,j} \xi_{ij}$$

subject to the constraints

$$\forall_{i=1}^{n}, \forall_{j=1}^{T_i}, \forall \bar{y} \in \mathcal{Y}_T(x_i) \quad \langle \mathbf{w}, \Phi(x_i, y_i) \rangle + \xi_{ij} \geq \langle \mathbf{w}, \Phi(x_i, \bar{y}) \rangle + \Delta_{total}(y_i, \bar{y}).$$

Note that Optimization Problem 8.2 has at most $\sum_{i=1}^{n} (T_i)^2$ constraints and can efficiently be solved with standard QP-solving techniques.

8.4 Experimental Results

In this section we evaluate the performance and benefit of batch detection on a collection of emails. We compare our learning methods with the iterative learning procedure for supervised clustering by Finley and Joachims (2005)

and perform an error analysis. We evaluate how the identification of email batches can actually support the classification of emails as spam or non-spam. Furthermore, we assess the execution time of the presented decoding methods. Quadratic programs are solved with CPLEX.

8.4.1 Email Batch Data

Email batch detection is performed at a mail transfer agent that processes a dense stream of messages. Standard email collections such as the Enron corpus or the TREC spam collection are collected from final recipients and therefore exhibit different characteristics. A mail transfer agent experiences many large batches over a short period of time. Existing spam corpora were harvested over a longer period from clients and contain fewer and more scattered copies of each batch. We therefore create an email corpus that reflects the characteristics of an email stream, but remedies the obvious privacy concerns that would arise from simply recording an email stream at a mail transfer agent. We do record the email stream for a short period of time, but only extract spam messages from this record. We randomly insert non-spam messages from the Enron collection and batches of newsletters. We remove the headers except for the sender address, MIME part information, and the header size.

The final corpus contains 2,000 spam messages, 500 Enron messages, and 500 newsletters (copies of 50 distinct newsletters). We manually group these emails into 136 batches with an average of 17.7 emails, and 598 remaining singleton mails. We implement 47 feature functions. They include the TFIDF similarity, equality of sender, equality of the MIME type, and differences in letter-bigram-counts.

We design a cross-validation procedure such that no elements of the same newsletter or spam batch occur in both the training and test set at any time. To this end, we construct each test set by using one non-singular batch, and filling the test sample with singletons and emails of other batches to a total size of 100. Batches with more than 50 emails are divided over several test sets, to ensure a reasonable mixture of emails from the test batch and other emails. Overall, there are 153 test sets. For each of these test sets, nine training sets $\mathcal{X}_1, \ldots, \mathcal{X}_9$ are generated by sampling randomly from the remaining emails, excluding emails from the test batch in case of split test batches. All reported results are averaged over the results from each of the 153 training/test combinations.

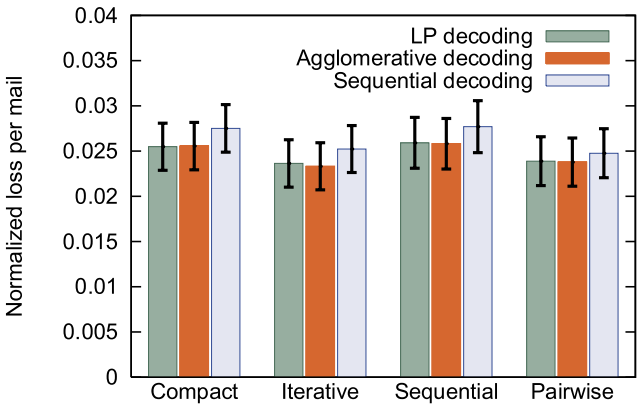


Figure 8.1: *Average loss for window size $T = 100$.*

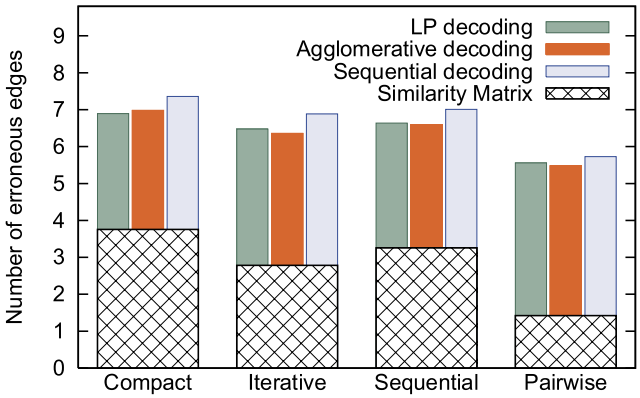


Figure 8.2: *Fraction of the loss induced by the learning algorithm (similarity matrix) and the decoding.*

8.4.2 Batch Identification

We compare the parameter vectors obtained by four strategies. Parameters are estimated by solving Optimization Problem 8.1 (compact), solving Optimization Problem 8.2 (sequential), and by using the iterative training algorithm of Finley and Joachims (2005) (iterative). As an additional baseline, we train a pairwise classifier (pairwise) as follows: Each pair of emails within a set constitutes a training example, with label $+1$ if they belong to the same cluster, and -1 otherwise. A linear SVM is trained on these pairs, and the weight vector is directly used as the parameter of the similarity measure. The final clustering is then obtained by one of the decoding strategies, using the similarity matrix obtained from pairwise learning. Note that the pairwise classifier is identical to PCC in Finley and Joachims (2005).

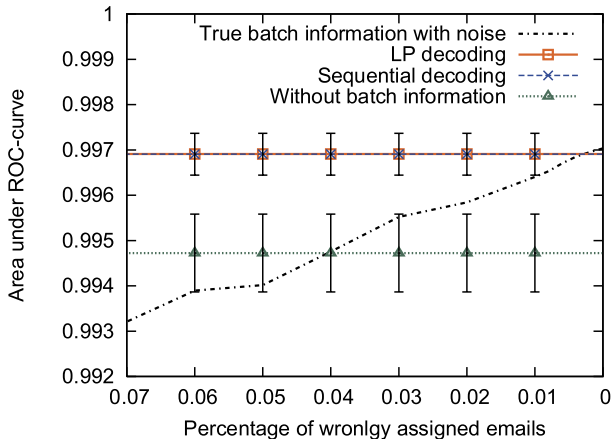


Figure 8.3: *Classification accuracy with batch information.*

Though three of the four optimization problems refer to a specific decoding strategy, we evaluate each of them with every decoder for comparison. We study three decoders: the LP decoder, the sequential decoder (Decoding Strategy 8.1), and the greedy agglomerative clustering described in Finley and Joachims (2005). Figure 8.1 shows the average normalized loss per mail of these combinations with standard errors. For this problem, there are no significant differences between either of these training and decoding methods. The sequential decoder operates under the constraint of linearity, and it would be plausible to assume that it incurs a higher loss than the LP decoding on average. The data suggests that this *might* be the case, but the difference is at most slight and by no means significant.

Figure 8.2 gives more insight into the characteristics of the compared methods. On the y-axis, the number of disagreeing edges with respect to the true clustering is depicted. The hatched areas indicate the number of disagreements between the true clustering and the signs of the similarity matrix induced by the weight vector and the pairwise features. The similarity matrix serves as input to the decoder; the decoder transforms it into a consistent partitioning. The bars indicate the numbers of incorrect edges after clustering.

It is apparent that the simplest learning method, pairwise learning, leads to the fewest wrong edges before clustering, but the induced similarity matrix is furthest away from being a consistent partitioning. This corresponds to the intuition that the training constraints of pairwise learning refer to individual links instead of the entire partitioning. The iterative algorithm leads to similarity matrices which are significantly nearer to a consistent clustering (i.e., the bars are shorter). The similarity measures learned by the compact

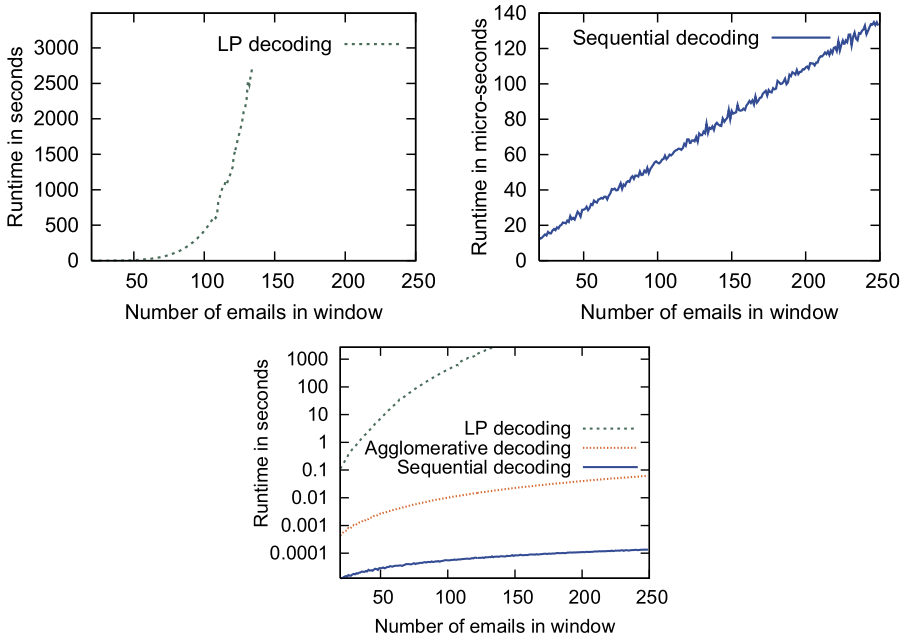


Figure 8.4: *Computation time for adding one email depending on window size.*

optimization problems lead to a similarity matrix with still more disagreeing edges, while yielding comparable error rates after decoding. This indicates that the decoding step has to resolve fewer inconsistencies, making it more robust to approximations.

8.4.3 Classification Using Batch Information

We evaluate how the classification of emails as spam or non-spam benefits from the identification of batches. As a baseline, we train a linear support vector machine with the word-counts of the training emails as features. We remove all email header information except for the subject line in order to eliminate artefacts from the data collection procedure.

We construct a collective filter that sums up the word counts of all emails in a batch, and includes four additional features: the size of the batch, a binary feature indicating whether the batch is larger than one, a binary feature indicating whether the subject of all emails in the batch is identical, and a binary feature indicating whether the sender address of all emails in the batch is identical. This results in all emails within the same batch having the same feature representation.

We examine how the classification performance is affected by the batch

detection. As an upper bound, we investigate the performance of the collective classifier given perfect clustering information, based on the manual clustering. In addition, we assess how sensitive the benefit of collective classification is with respect to the accuracy of the clustering. In the setting of clustering with noise, each email is collectively classified in a cluster that contains an increasing number of wrongly clustered emails.

Figure 8.3 shows the area under the ROC curve (AUC) for the classifiers under investigation. The performance of the collective classifier based on a perfect clustering can be seen on the right hand side of the graph (ideal clustering at 0% noise). The difference between the collective classification based on a perfect clustering and a classification based on the inferred clusterings is not significant. The collective classifiers perform indistinguishably well; the sequential and LP decoder perform alike. We can see that using ideal batch information, the risk of misclassification (1 - AUC) is reduced by 43.8%, while with non-ideal batch information obtained through approximate clustering a reduction of 41.4% is still achieved. Even though the AUC of the baseline appears high already, a 40% reduction of the risk in spam filtering is a substantial improvement!

8.4.4 Clustering Runtime

Efficiency is an important aspect in clustering on streams, especially in identifying spam batches. The window size has to be sufficiently large to contain at least one representative of each currently active batch. The time required to cluster one additional email depending on the window size is therefore a crucial criterion for selecting an appropriate clustering method.

Figure 8.4 illustrates the observed time required for processing an email by LP decoding and sequential decoding with respect to the window size. While the computation time of the LP approximation grows at least cubically, the time for an incremental update for a single email with sequential decoding grows only linearly. Due to the different time scales of the two methods (note that the upper right graph shows micro-seconds instead of seconds), we use a logarithmic time scale to plot the curves in a single diagram (bottom graph).

8.5 Conclusions

We devised a sequential clusering algorithm and two integrated formulations for learning a similarity measure to be used with correlation clustering. First, we derived a compact optimization problem based on the LP approximation to correlation clustering to learn the weights of the similarity measure. Start-

ing from the assumption that decisions for already processed emails cannot be reconsidered, we devised an efficient clustering algorithm that is linear in the number of emails in the window. From this algorithm we derived a second integrated method for learning the weight vector.

Our empirical results indicate that there are no significant differences between the learning or decoding methods in terms of accuracy. Yet the integrated learning formulations optimize the weight vector more directly to yield consistent partitionings. Using the batch information obtained from decoding with the learned models, email spam classification performance increases substantially over the baseline with no batch information. The efficiency of the sequential clustering algorithm makes supervised batch detection at enterprise-level scales, with millions of emails per hour and thousands of recent emails as reference, feasible.

Chapter 9

Conclusions

This thesis dealt with semi-supervised prediction models for structured output spaces. We lifted the techniques of classical semi-supervised learning to the structured setting and devised novel, semi-supervised algorithms based on state-of-the-art approaches in structural learning. The presented approaches either implement the consensus maximization principle or rely on a cluster assumption in the data. Empirical results showed that the semi-supervised algorithms outperform appropriate baselines in many application areas, including multi-class classification, named entity recognition, and natural language parsing.

We introduced the co-learning setting with univariate function approximation as a special case of learning in structured output spaces. In contrast to many other multi-view approaches, our formulation required only the unlabeled data to be equal in all views whereas the labeled examples may differ from view to view. We devised an exact closed form solution to co-regularized least squares regression that scales cubic in the number of labeled and unlabeled instances. Since being cubic in the unlabeled sample size is not a desirable property, we proposed an approximate, semi-parametric coRLSR that scales only linear in the number of unlabeled instances. Semi-parametric coRLSR also provides a solution in closed form. Additionally, we derived a distributed optimization scheme where participants keep their labeled examples private and only have to agree on unlabeled data. We provided results on convergence properties of the distributed optimization where only predictions on unlabeled examples need to be shared among the participants. Empirically, we showed that both semi-supervised algorithms significantly outperform fully-supervised RLSR. Although the exact, non-parametric coRLSR led to lower root mean squared errors than the semi-parametric approximation, the latter can be trained in large-scale, allowing for a further decrease in errors.

We generalized the consensus maximization principle to learning in structured output spaces and devised the semi-supervised co-perceptron. Applications to named entity recognition tasks showed that random features splits perform significantly better than splitting the features into token and surface clue views. Nevertheless, even for the weak split, co-perceptron outperformed its fully supervised counterpart. We derived co-support vector machines by taking a large margin approach in joint input-output space. CoSVMs allow the inclusion of arbitrary loss functions and enforce confident predictions by maximizing the margin. We derived primal and dual optimization problems for 1- and 2-norm coSVMs with slack rescaled losses. Empirical results showed that coSVMs outperform every single-view baseline method in multiclass, named entity recognition, and natural language parsing tasks, where we applied various loss functions.

To explore the benefit of explicitly assuming a cluster structure in the data, we studied structural transductive learning. Lifting the classical transductive approach to the structured domain in a one-to-one relation renders the optimization problem intractable for practical applications. As a remedy, we translated the constrained and non-differentiable optimization problem into an unconstrained and differentiable objective function. Although the ∇TSVM criterion is no longer convex, it now can be efficiently optimized by gradient-based techniques. As a by-product, we also devised unconstrained and differentiable variants of structural SVM and structural coSVM. Empirically, unconstrained optimization led to a significant decrease in execution time compared to standard approaches solved by quadratic programming. In terms of performance, ∇TSVM failed to make use of the unlabeled data in many settings due to poor local minima found by non-convex optimization and due to inappropriate cluster assumptions. From the direct comparison with multi-view support vector machines we conclude that co-learning represents an appealing alternative to cluster-based semi-supervised learning in settings where the latter is known to be inappropriate.

We showed in a case study on email batch detection the benefit of exploiting the cluster structure when the task at hand preserves a cluster assumption. We thus translated the problem into a supervised clustering task and derived a solution based on a separating margin. The resulting support vector machine used loss-augmented inference to implement a relaxed variant of correlation clustering. The cubic number of constraints rendered the optimization intractable, however, the initial solution did not exploit the streaming nature of the data. We devised a linear time approximation by effectively utilizing the sequential data. The sequential approach is shown to perform as well as all other baseline methods but its execution time makes it uniquely scalable for practical applications. Moreover, we showed that hav-

ing the batch information by the sequential clustering can lead to a significant reduction of the spam misclassification risk.

Building on this line of research there are many problems that are promising candidates for future investigations and extensions of the presented results. An exciting direction of semi-supervised structured learning deals with missing values in input and/or output structures. Exemplary applications include sequence labeling tasks with only partially annotated input and output sequences, network completion tasks aiming at amending incomplete link structure, and alignment tasks such as machine translation. These tasks do not necessarily exhibit a one-to-one relation between observations and labels. For instance, in machine translation, a token in the source language may be mapped to several tokens in the target language and vice versa. Moreover, several correspondences between observations and labelings may not be observed in the training data. The limit is, of course, a fully unsupervised setting, where no information about true labelings is available.

Another related area is learning under covariate shift. In this setting, training and test data are drawn from different distributions. Recently, this setting has gained more and more attention, however, previous work focuses on classical supervised learning with univariate response variables. Applying the lessons learned to the structured domain is promising in the presence of scarce and expensive labeled data. Moreover, learning under covariate shift is also highly connected to transfer learning that might be of substantial interest in future applications. The underlying idea is to train a model on one domain and apply it to another.

Last but not least, discriminative structured prediction models are still computationally expensive – and the proposed semi-supervised extensions in particular. Techniques to speed up training processes significantly and allowing for large-scale experiments are therefore of great interest for achieving high predictive accuracy.

Bibliography

S. Abney. Bootstrapping. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2002.

Y. Altun, M. Johnson, and T. Hofmann. Discriminative learning for label sequences via boosting. In *Advances in Neural Information Processing Systems*, 2003a.

Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Markov support vector machines. In *Proceedings of the International Conference on Machine Learning*, 2003b.

Y. Altun, T. Hofmann, and A. J. Smola. Gaussian process classification for segmenting and annotating sequences. In *Proceedings of the International Conference on Machine Learning*, 2004a.

Y. Altun, T. Hofmann, and A.J. Smola. Exponential families for conditional random fields. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2004b.

Y. Altun, D. McAllester, and M. Belkin. Maximum margin semi–supervised learning for structured variables. In *Advances in Neural Information Processing Systems*, 2006.

Y. Altun, T. Hofmann, and I. Tsochantaridis. SVM learning for interdependent and structured output spaces. In *Machine Learning with Structured Outputs*, 2007.

S. Baluja. Probabilistic modeling for face orientation discrimination: Learning from labeled and unlabeled data. In *Advances in Neural Information Processing Systems*, 1998.

N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1–3):89–113, 2004.

Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, 2002.

A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning distance functions using equivalence relations. In *Proceedings of the International Conference on Machine Learning*, 2003.

S. Barnett. *Matrix Methods for Engineers and Scientists*. MacGraw-Hill, 1979.

M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi–supervised learning on large graphs. In *Proceedings of the Conference on Learning Theory*, 2004.

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

K. Bennet and A. Demiriz. Semi–supervised support vector machines. In *Advances in Neural Information Processing Systems*, 1998.

D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

S. Bickel and T. Scheffer. Multi–view clustering. In *Proceedings of the IEEE International Conference on Data Mining*, 2004.

S. Bickel and T. Scheffer. Dirichlet-enhanced spam filtering based on biased samples. In *Advances in Neural Information Processing Systems*, 2007.

M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2003.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

C. M. Bishop. *Pattern Recognition in Machine Learning*. Springer, 2006.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co–training. In *Proceedings of the Workshop on Computational Learning Theory*, 1998.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Workshop on Computational Learning Theory*, 1992.

Z. Bosnic, I. Kononenko, M. Robnic-Sikonja, and M. Kukar. Evaluation of prediction reliability in regression using the transduction principle. In *The IEEE Region 8 EUROCON 2003: Computer as a Tool*, 2003.

L. Bouttou, C. Cortes, J. Denker, I. Guyon H. Drucker, L. Jackel, Y. le Cun, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparision of classifier methods: A case study in handwriting digit recognition. In *Proceedings of the International Conference on Pattern Recognition*, 1994.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Camebridge University Press, 2004.

U. Brefeld and T. Scheffer. Co–EM support vector learning. In *Proceedings of the International Conference on Machine Learning*, 2004.

S. Brin, J. Davis, and H. García-Molina. Copy detection mechanisms for digital documents. In *Proceedings of the International Conference on Management of Data*, 1995.

M. Brückner and W. Dilger. A soft Bayes perceptron. In *Proceedings of the International Joint Conference on Neural Networks*, 2005.

O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19:1155–1178, 2006.

O. Chapelle and A. Zien. Semi–supervised classification by low density separation. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, 2005.

O. Chapelle, V. Vapnik, and J. Weston. Transductive inference for estimating values of functions. In *Advances in Neural Information Processing Systems*, 1999.

O. Chapelle, M. Chi, and A. Zien. A continuation method for semi–supervised SVMs. In *Proceedings of the International Conference on Machine Learning*, 2006a.

O. Chapelle, B. Schölkopf, and A. Zien. *Semi–supervised Learning*. MIT Press, 2006b.

M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005.

Z. Chi. Statistical properties of probabilistic context–free grammars. *Computational Linguistics*, 25(1):131–160, 1999.

F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

S. Clark and J. R. Curran. Parsing the wsj using ccg and log-linear models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2004.

M. Collins. Discriminative reranking for natural language processing. In *Proceedings of the International Conference on Machine Learning*, 2000.

M. Collins. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2002.

M. Collins and N. Duffy. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems*, 2002.

M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.

R. Collobert and S. Bengio. Links between perceptrons, MLPs and SVMs. In *Proceedings of the International Conference on Machine Learning*, 2004.

D. Cooper and J. Freeman. On the asymptotic improvement in the outcome of supervised learning provided by additional nonsupervised learning. *IEEE Transactions on Computers*, C-19:1055–1063, 1970.

J. Cooper, A. Coden, and E. Brown. A novel method for detecting similar documents. In *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2002a.

J. Cooper, A. Coden, and E. Brown. Detecting similar documents using salient terms. In *Proceedings of the International Conference on Information and Knowledge Management*, 2002b.

C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20: 273–297, 1995.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.

F. Cozman, I. Cohen, and M. Cirelo. Semi–supervised learning of mixture models. In *Proceedings of the International Conference on Machine Learning*, 2003.

K. Crammer and Y. Singer. On the algorithmic implementation of multi-class kernel-based vector machines. *Journal of Machine Learning Research*, 2: 265–292, 2001.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning algorithms*. Cambridge University Press, 2000.

E. Damiani, S. de Capitani di Vimercati, S. Paraboschi, and P. Samarati. P2P-based collaborative spam detection and filtering. In *Proceedings of the International Conference on Peer-to-Peer Computing*, 2004.

J. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43:1470–1480, 1972.

S. Dasgupta, M. Littman, and D. McAllester. PAC generalization bounds for co–training. In *Proceedings of Neural Information Processing Systems*, 2001.

A. P. Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society, Series B*, 41:1–31, 1979.

V. de Sa. Learning classification with unlabeled data. In *Proceedings of Neural Information Processing Systems*, 1994.

E. D. Demaine and N. Immorlica. Correlation clustering with partial information. In *Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization Problems and International Workshop on Randomization and Approximation Techniques in Computer Science*, 2003.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

T. G. Dietterich, A. Ashenfelter, and Y. Bulatov. Training conditional random fields via gradient tree boosting. In *Proceedings of the International Conference on Machine Learning*, 2004.

T.G. Dietterich. Machine learning for sequential data: A review. In *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, 2002.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd edition)*. John Wiley, 2001.

B. Efron. The geometry of exponential families. *The Annals of Statistics*, 6(2):362–376, 1978.

D. Emanuel and A. Fiat. Correlation clustering – minimizing disagreements on arbitrary weighted graphs. In *Proceedings of the Annual European Symposium on Algorithms*, 2003.

J. D. R. Farquhar, D. Hardoon, H. Meng, J. Shawe-Taylor, and S. Szedmák. Two view learning: SVM–2K, theory and practice. In *Advances in Neural Information Processing Systems*, 2006.

T. Finley and T. Joachims. Supervised clustering with support vector machines. In *Proceedings of the International Conference on Machine Learning*, 2005.

G. D. Forney. The Viterbi algorithm. *Proceedings of IEEE*, 61(3):268–278, 1973.

Y. Freund and R. E. Shapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

Y. Freund, R. Iyer, R. E. Shapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *Proceedings of the International Conference on Machine Learning*, 1998.

K. Fukunga. *Introduction to Statistical Pattern Analysis*. Academic Press, 1990.

G. Fung and O. L. Mangasarian. Proximal support vector machines. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2001.

J. Fürnkranz. Round robin ensembles. *Intelligent Data Analysis*, 7(5):385–404, 2003.

S. Geman and M. Johnson. Dynamic programming for parsing and estimation of stochastic unification-based grammars. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2002.

C. Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242, 2001.

R. Ghani. Combining labeled and unlabeled data for multiclass text categorization. In *Proceedings of the International Conference on Machine Learning*, 2002.

C. L. Giles, G. M. Kuhn, and R. J. Williams. Special issue on dynamic recurrent neural networks. *IEEE Transactions on Neural Networks*, 5(2), 1994.

S. Goldman and Y. Zhou. Enhancing supervised learning with unlabeled data. In *Proceedings of the International Conference on Machine Learning*, 2000.

A. Gray and M. Haahr. Personalised, collaborative spam filtering. In *Proceedings of the Conference on Email and Anti-Spam*, 2004.

S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data Engineering.*, 15(3):515–528, 2003.

J. Hakenberg, S. Bickel, C. Plake, U. Brefeld, H. Zahn, L. Faulstich, U. Leser, and T. Scheffer. Systematic feature evaluation for gene name recognition. *BMC Bioinformatics*, 6(1):S9, 2005.

J. M. Hammersley and P. E. Clifford. Markov random fields on finite graphs and lattices. Unpublished manuscript, 1971.

S. Har-Peled, D. Roth, and D. Zimak. Constraint classification for multi–class classification and ranking. In *Advances in Neural Information Processing Systems*, 2002.

D. Haussler. Convolution kernels on discrete structures. Technical Report UCS-CRL-99-10, University of California, Santa Cruz, 1999.

R. Herbrich. *Learning kernel classifiers: Theory and Algorithms*. MIT Press, 2002.

R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *In Proceedings of the International Conference on Articial Neural Networks*, 1999.

M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49:409–436, 1952.

A. Jakulin and I. Bratko. Testing the significance of attribute interactions. In *Proceedings of the International Conference on Machine Learning*, 2004.

E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4):620–630, 1957.

T. Joachims. *Learning to Classify Text using Support Vector Machines*. Kluwer, 2002.

T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the International Conference on Machine Learning*, 2003.

T. Joachims. A support vector method for multivariate performance measures. In *Proceedings of the International Conference on Machine Learning*, 2005.

T. Joachims. Training linear SVMs in linear time. In *Proceedings of the Conference on Knowledge Discovery and Data Mining*, 2006.

T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the International Conference on Machine Learning*, 1999a.

T. Joachims. Making large-scale SVM learning practical. In *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1999b.

T. Joachims and J. Hopcroft. Error bounds for correlation clustering. In *Proceedings of the International Conference on Machine Learning*, 2005.

M. Johnson. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632, 1998.

M. Johnson, S. Geman, S. Canon, Z. Chi, and S. Riezler. Estimators for stochastic "unification-based" grammars. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 1999.

M. I. Jordan. Serial order: A parallel distributed processing approach. Technical Report 8604, Institute for Cognitive Science, University of California, San Diego, 1987.

M. I. Jordan and T. J. Sejnowski. *Graphical Models: Foundations of neural computation*. MIT Press, 2001.

B. Juang and L. Rabiner. Hidden Markov models for speech recognition. *Technometrics*, 33:251–272, 1991.

T. Kasami. An efficient recognition and syntax algorithm for context-free languages. Technical Report AFCLR-65-758, Air Force Cambridge Research Laboratory, Bedford, MA, 1965.

S. S. Keerthi and D. M. DeCoste. A modified finite Newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, 6:341–361, 2005.

J. E. Kelley. The cutting-plane method for solving convex programs. *Journal of the Scociety for Industrial Applied Mathematics*, 8:703–712, 1960.

J. G. Kemeny, J. L. Snell, and A. W. Knapp. *Denumerable Markov chains (second edition)*. Springer, 1976.

M. Kockelkorn, A. Lüneburg, and Tobias Scheffer. Using transduction and multi–view learning to answer emails. In *Proceedings of the European Conference on Principle and Practice of Knowledge Discovery in Databases*, 2003.

A. Kolcz, A. Chowdhury, and J. Alspector. The impact of feature selection on signature-driven spam detection. In *Proceedings of the Conference on Email and Anti-Spam*, 2004.

U. Kressel. Pairwise classification and support vector machines. In *Advances in Kernel Methods – Support Vector Learning*, 1999.

B. Krishnapuram, D. Williams, Y. Xue, A. Hartemik, and L. Carin. On semi–supervised classification. In *Advances in Neural Information Processing Systems*, 2004.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, 2001.

J. Lafferty, X. Zhu, and Y. Liu. Kernel conditional random fields: representation and clique selection. In *Proceedings of the International Conference on Machine Learning*, 2004.

S. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

Q. V. Le, A. J. Smola, T. Gärtner, and Y. Altun. Transductive gaussian process regression with automatic model selection. In *Proceedings of the European Conference on Machine Learning*, 2006.

C. Lee, S. Wang, F. Jiao, R. Greiner, and D. Schuurmans. Learning to model spatial dependency: Semi–supervised discriminative random fields. In *Advances in Neural Information Processing Systems*, 2007.

B. Leskes. The value of agreement, a new boosting algorithm. In *Proceedings of the Conference on Learning Theory*, 2005.

D. J. C. MacKay. Introduction to Gaussian processes. *Neural Networks and Machine Learning*, 168:133–165, 1998.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19:313–330, 1993.

M. P. Marcus, G. Kim, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. The Penn treebank: Annotating predicate argument structure. In *ARPA Human Language Technology Workshop*, 1994.

D. McAllester, M. Collins, and F. Pereira. Case-factor diagrams for structured probabilistic modeling. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2004.

A. McCallum and K. Nigam. Employing EM in pool-based active learning for text classification. In *Proceedings of the International Conference on Machine Learning*, 1998.

A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the International Conference on Machine Learning*, 2000.

M. Meila. Comparing clusterings by the variation of information. In *Proceedings of the Conference on Computational Learning Theory*, 2003.

H. Meng, J. Shawe-Taylor, S. Szedmák, and J. D. R. Farquhar. Support vector machine to synthesise kernels. In *Deterministic and Statistical Methods in Machine Learning*, 2004.

Y. Miyao and J. Tsujii. Maximum entropy estimation for feature forests. In *Proceedings of Human Language Technology Conference*, 2002.

D. Mladenic. Learning word normalization using word suffix and context from unlabeled data. In *Proceedings of the International Conference on Machine Learning*, 2002.

K.P. Murphy, Y. Weiss, and M.I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of Uncertainty in Aritifical Intelligence*, 1999.

I. Muslea, C. Knoblock, and S. Minton. Active + semi–supervised learning = robust multi–view learning. In *Proceedings of the International Conference on Machine Learning*, 2002.

D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998.

M. Ng, E. Chan, M. So, and W. Ching. A semi–supervised regression model for mixed numerical and categorical variables. *Pattern Recognition*, 40(6): 1745–1752, 2007.

N. Nguyen and Y. Guo. Comparisons of sequence labeling algorithms and extensions. In *Proceedings of the International Conference on Machine Learning*, 2007.

K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co–training. In *Proceedings of Information and Knowledge Management*, 2000.

Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

A. B. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, 1962.

C. Ordonez. Clustering binary data streams with k-means. In *Proceedings of the ACM SIGMOD workshop on Research Issues in Data Mining and Knowledge Discovery*, 2003.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Advances in Large Margin Classifiers*, 2000.

T. Poggio, S. Mukherjee, R. Rifkin, A. Rakhlin, and A. Verri. b. In *Proceedings of the Conference on Uncertainty in Geometric Computations*, 2001.

A. Pozdnoukhov and S. Bengio. Semi–supervised kernel methods for regression estimation. In *Proceedings of the IEEE International Conference on Acoustic, Speech, and Signal Processing*, 2006.

L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:622–626, 1971.

B. Raskutti, H. Ferra, and A. Kowalczyk. Combining clustering and co–training to enhance text classification using unlabeled data. In *Proceedings of the SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. The MIT Press, 2006.

G. Rätsch and S. Sonnenburg. Large scale hidden semi-Markov SVMs. In *Advances in Neural Information Processing Systems*, 2007.

R. M. Rifkin. *Everything Old is new again: A fresh Look at Historical Approaches to Machine Learning*. PhD thesis, MIT, 2002.

E. S. Ristad and P. N. Yianilos. Learning string edit distance. In *Proceedings of the International Conference on Machine Learning*, 1997.

H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

D. Roth and W. Yih. Integer linear programming inference for conditional random fields. In *Proceedings of the International Conference on Machine Learning*, 2005.

E. F. Tjong Kim Sang. Introduction to the CoNLL-2002 shared task: language-independent named entity recognition. In *Proceedings of the Conference on Computational Natural Language Learning*, 2002.

C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the International Conference on Machine Learning*, 1998.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

B. Schölkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 1995.

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *Proceedings of the Annual Conference on Learning Theory*, 2001.

D. Schuurmans, F. Southey, D. Wilkinson, and Y. Guo. Metric–based approaches for semi–supervised regression and classification. In *Semi–Supervised Learning*, 2006.

A. Schwaighofer and V. Tresp. Transductive and inductive methods for approximate Gaussian process regression. In *Advances in Neural Information Processing Systems*, 2003.

R. Schwarz and Y. L. Chow. The n-best algorithm: An efficient and exact procedure for finding the n most likely hypotheses. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 1990.

M. Seeger. Learning with labeled and unlabeled data. (technical report, University of Edinburgh, 2001.

T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronounce english text. *Journal of Complex Systems*, 1(1):145–168, 1987.

S. Seo, M. Wallat, T. Graepel, and K. Obermayer. Gaussian process regression: Active data selection and test point rejection. In *Proceedings of the International Joint Conference on Neural Networks*, 2000.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. Technical Report CIS TR MS-CIS-02-35, University of Pennsylvania, 2003.

B. Shahshahani and D. Landgrebe. The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon. *IEEE Transactions on Geoscience and Remote Sensing*, 32: 1087–1095, 1994.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub–gradient solver for SVM. In *Proceedings of the International Conference on Machine Learning*, 2007.

L. Shen, A. Sarkar, and A. K. Joshi. Using ltag based features in parse reranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2003.

H. Shin, N. J. Hill, and G. Rätsch. Graph based semi–supervised learning with sharper edges. In *Proceedings of the European Conference on Machine Learning*, 2006.

V. Sindhwani, P. Niyogi, and M. Belkin. A co–regularized approach to semi–supervised learning with multiple views. In *Proceedings of the ICML Workshop on Learning with Multiple Views*, 2005a.

V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: From transductive to semi–supervised learning. In *Proceedings of the International Conference on Machine Learning*, 2005b.

W. Skut, B. Krenn, T. Brants, and H. Uszkoreit. An annotation scheme for free word order languages. In *Proceedings of the Conference on Applied Natural Language Processing*, 1997.

A. J. Smola and I. R. Kondor. Kernels and regularization on graphs. In *Proceedings of the Annual Conference on Computational Learning Theory*, 2003.

A. J. Smola and B. Schölkopf. A tutorial on support vector regression. Technical Report NC2-TR-1998-030, 1998, NeuroCOLT2, 1998.

J. A. K. Suykens. Least squares support vector machines for classification and nonlinear modelling. *Neural Network World*, 10, 2000.

C. Swamy. Correlation clustering: Maximizing agreements via semidefinite programming. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 2004.

S. Szedmák and J. Shawe-Taylor. Synthesis of maximum margin and multi–view learning using unlabeled data. In *Proceedings of the European Symposium on Artificial Neural Networks*, 2006.

B. Taskar, C. Guestrin, and D. Koller. Max–margin Markov networks. In *Advances in Neural Information Processing Systems*, 2004a.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. Max–margin parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2004b.

B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *Proceedings of the International Conference on Machine Learning*, 2005.

A. N. Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Math. Dokl.*, 4:1035–1038, 1963.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the International Conference on Machine Learning*, 2004.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

J. Verbeek and N. Vlassis. Gaussian fields for semi–supervised regression and correspondence learning. *Pattern Recognition*, 39(10):1864–1875, 2006.

S.V.N. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, and K. Murphy. Accelerated training of conditional random fields with stochastic meta-descent. In *Proceedings of the International Conference on Machine Learning*, 2006.

K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the International Conference on Machine Learning*, 2001.

G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, 1990.

J. Weston and C. Watkins. Multi–class support vector machines. Technical Report CSD-TR-98-04, Department of Computer Sciences, Royal Holloway, University of London, 1998.

C. K. I. Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In *Learning and Inference in Graphical Models*, 1999.

E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems*, 2002.

L. Xu, D. Wilkinson, F. Southey, and D. Schuurmans. Discriminative unsupervised learning of structured predictors. In *Proceedings of the International Conference on Machine Learning*, 2006.

Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1/2):69–90, 1999.

D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 1995.

A. Yeh, A. Morgan, M. Colosimo, and L. Hirschman. BioCreative task 1A: Gene mention finding evaluation. *BMC Bioinformatics 2005*, 6(1): S2, 2005.

D. H. Younger. Recognition and parsing of context free languages in time n^3. *Information and Control*, 10(2):189–208, 1967.

D. Zhou, B. Schölkopf, and T. Hofmann. Semi–supervised learning on directed graphs. In *Advances in Neural Information Processing Systems*, 2005.

F. Zhou, L. Zhuang, B. Y. Zhao, L. Huang, A. D. Joseph, and J. Kubiatowicz. Approximate object location and spam filtering on peer-to-peer systems. In *Proceedings of Middleware*, 2003.

Y. Zhou and S. Goldman. Democratic co–learning. In *Proceedings of the International Conference on Tools with Artificial Intelligence*, 2004.

Z.-H. Zhou and M. Li. Semi–supervised regression with co–training. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2005.

X. Zhu. Semi–supervised learning in literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi–supervised learning using gaussian fields and harmonic functions. In *Proceedings of the International Conference on Machine Learning*, 2003.

Appendix A

Univariate Learning Algorithms

The presented framework of learning in joint input-output spaces covers classical learning algorithms as special cases. The model $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ returns a real value for any input output pair (x, y) detailing how good output y fits to the input x. Thus, for a given input, f imposes a ranking on the output space. For instance, identifying input x as a query and $\mathcal{Y}$ as a set of documents, f may be used directly for information retrieval tasks.

Moreover, the proposed framework also contains classical learning scenarios such as regression and classification as special cases. On one hand, the objective

$$\hat{y} = \operatorname*{argmax}_{\bar{y} \in \mathcal{Y}(x)} f(x, \bar{y}) \tag{A.1}$$

naturally reduces to a regression setting when $\mathcal{Y}(x) = \mathbb{R}$ for all $x \in \mathcal{X}$, that is, returning the most likely function value at input x. On the other hand, Equation A.1 reduces to binary classification when the output space contains precisely two elements, that is, $\mathcal{Y} = \{+1, -1\}$.

In this section, we review classical supervised learning algorithms allowing for a dual representation that depends only on inner products in some feature space. We assume that every input example x_i is translated into a d-dimensional feature vector $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,d})^\mathsf{T}$. The algorithms are extended to semi-supervised learning and/or to deal with structured output spaces in previous chapters. We begin with a derivation of regularized least squares regression in Section A.1 followed by two binary classification algorithms, the perceptron (Section A.2) and support vector machines (Section A.3).

A.1 Regularized Least Squares Regression

In this section, we introduce regularized least squares regression (RLSR) (Rifkin, 2002) from a Bayesian point of view. Regularized least squares regression is also referred to as ridge regression (Saunders et al., 1998), least squares support vector machines (Suykens, 2000), and proximal support vector machines (Fung and Mangasarian, 2001).

Function approximation from a Bayesian perspective frequently implies a standard linear regression model with Gaussian noise

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle, \quad y = f(\mathbf{x}) + \epsilon. \tag{A.2}$$

We call $\mathbf{x} \in \mathbb{R}^d$ the observed input vectors, $y \in \mathbb{R}$ the real valued outputs (targets), and $\mathbf{w}$ the vector of weights. Equation A.2 implies that the weight vector $\mathbf{w}$ is normal to the seeked regression plane. From A.2 it is also clear that such a regression hyperplane has to pass through the origin. However, this is a strong assumption on the form of the model and if this is inappropriate, the model may perform poorly. We can overcome this drawback by augmenting every input vector $\mathbf{x}$ by a constant element $c \in \mathbb{R}$ and define $\tilde{\mathbf{x}} = (c, x_1, \ldots, x_d)^\mathsf{T}$. In the following we continue with the notation $\mathbf{x}$ wether we use augmented input vectors or not.

The noise term in Equation A.2 is assumed to be iid Gaussian with zero mean and variance σ^2

$$\epsilon \sim N(0, \sigma^2). \tag{A.3}$$

Solving Equation A.2 for ϵ implies that $y - f(\mathbf{x})$ is also iid Gaussian with zero mean and variance σ^2. That is, given n pairs $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ with $\mathbf{x}_i \in \mathcal{X}$ and $y \in \mathbb{R}$, the conditional likelihood of the targets $\mathbf{y} = (y_1, \ldots, y_n)^\mathsf{T}$ can be written as

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}_1, \ldots, \mathbf{x}_n; \mathbf{w}) &= \prod_{i=1}^{n} p(y_i|\mathbf{x}_i; \mathbf{w}) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ -\frac{(y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle)^2}{2\sigma^2} \right\} \\ &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left\{ -\frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 \right\}, \end{aligned} \tag{A.4}$$

where the input vectors are aggregated in the $n \times d$-matrix $\mathbf{X}$ with elements $[\mathbf{X}]_{ij} = x_{ij}$, such that the i-th row of $\mathbf{X}$ contains the i-th input vector $\mathbf{x}_i$. Notice that the likelihood only factorizes over the instances because of the independence assumption.

A simple way to find good parameters is maximizing the likelihood directly with respect to the weights $\mathbf{w}$ leading to the optimization problem

$$\mathbf{w}^\star = \operatorname*{argmax}_{\mathbf{w}} p(\mathbf{y}|\mathbf{x}_1, \ldots, \mathbf{x}_n; \mathbf{w}). \tag{A.5}$$

However, maximum likelihood will inevitably lead to poor generalization performance for high-dimensional problems. As seen in Section 2.1, a remedy is often achieved by regularization that we will incorporate in terms of a *prior* on the weights.

In a Bayesian approach we need to define a prior on the weights, expressing beliefs about parameters before looking at the data. We favor sparse models, having zero weights for redundant and irrelevant features and thus apply a zero mean Gaussian prior and the identity as covariance matrix

$$\mathbf{w} \sim N(\mathbf{0}, \mathbb{1}).$$

Note that this special choice of a covariance matrix implies an independency of the components of $\mathbf{w}$. According to *Bayes' Theorem*, the posterior distribution of the weights can be written in terms of likelihood and prior

$$p(\mathbf{w}|\mathbf{y}, \mathbf{x}_1, \ldots, \mathbf{x}_n) = \frac{p(\mathbf{y}|\mathbf{x}_1, \ldots, \mathbf{x}_n; \mathbf{w}) p(\mathbf{w})}{p(\mathbf{y}|\mathbf{x}_1, \ldots, \mathbf{x}_n)}.$$

The denominator is a normalization term given by the marginalized likelihood over all values of $\mathbf{w}$

$$p(\mathbf{y}|\mathbf{x}_1, \ldots, \mathbf{x}_n) = \int p(\mathbf{y}|\mathbf{x}_1, \ldots, \mathbf{x}_n; \mathbf{w}) p(\mathbf{w}) d\mathbf{w}.$$

However, for our purposes it suffices to compute the most likely weight vector. We thus drop the normalization and address our goal directly by applying the argument of the maximum which is also called the maximum a posteriori (MAP) estimator,

$$\operatorname*{argmax}_{\mathbf{w}} p(\mathbf{w}|\mathbf{y}, \mathbf{x}_1, \ldots, \mathbf{x}_n) = \operatorname*{argmax}_{\mathbf{w}} p(\mathbf{y}|\mathbf{x}_1, \ldots, \mathbf{x}_n; \mathbf{w}) p(\mathbf{w}).$$

The log-posterior distribution is proportional to

$$\begin{aligned} \ln p(\mathbf{w}|\mathbf{y}, \mathbf{x}_1, \ldots, \mathbf{x}_n) &\propto -\frac{n}{2} \ln (2\pi\sigma^2) - \frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 - \frac{n}{2} \ln(2\pi) - \frac{1}{2} \mathbf{w}^\mathsf{T} \mathbf{w} \\ &\propto -\frac{1}{\sigma^2} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 - \mathbf{w}^\mathsf{T} \mathbf{w}. \end{aligned}$$

In other words, under the assumptions of Gaussian noise and mutual independence of the weights, the maximizer of the posterior distribution of the

weights is equivalent to the minimizer of a regularized least squares regression,

$$\underset{\mathbf{w}}{\operatorname{argmax}}\, p(\mathbf{w}|\mathbf{y}, \mathbf{x}_1, \ldots, \mathbf{x}_n) = \underset{\mathbf{w}}{\operatorname{argmin}}\, \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \sigma^2\|\mathbf{w}\|^2, \tag{A.6}$$

where the variance of the noise acts as a regularization constant. According to the representer theorem we already know that the minimizer of Equation A.6 allows for the representation

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \cdot) \quad \Rightarrow \quad \mathbf{X}\mathbf{w} = \mathbf{K}\boldsymbol{\alpha}, \tag{A.7}$$

with $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^\mathsf{T}$ and kernel matrix $\mathbf{K}$ with elements $[\mathbf{K}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \langle \psi(\mathbf{x}_i), \psi(\mathbf{x}_j) \rangle$ defined by an arbitrary feature mapping $\psi(\mathbf{x})$. Plugging A.7 into A.6 leads to Equation A.8 that has to be minimized with respect to every α_i.

$$Q(\boldsymbol{\alpha}) = \|\mathbf{y} - \mathbf{K}\boldsymbol{\alpha}\|^2 + \sigma^2 \boldsymbol{\alpha}^\mathsf{T}\mathbf{K}\boldsymbol{\alpha} \tag{A.8}$$

Setting the derivative of Q with respect to vector $\boldsymbol{\alpha}$ to zero yields the minimizer of A.6 and A.8 in terms of $\boldsymbol{\alpha}$. Using the symmetry of $\mathbf{K}$, we have

$$\begin{aligned} \frac{\partial Q}{\partial \boldsymbol{\alpha}} = -2\mathbf{K}\mathbf{y} + 2\mathbf{K}^\mathsf{T}\mathbf{K}\boldsymbol{\alpha} + 2\sigma^2\mathbf{K}\boldsymbol{\alpha} &\stackrel{!}{=} \mathbf{0} \\ 2\mathbf{K}^\mathsf{T}\mathbf{K}\boldsymbol{\alpha} + 2\sigma^2\mathbf{K}\boldsymbol{\alpha} &= 2\mathbf{K}\mathbf{y} \\ \left(\mathbf{K} + \mathbb{1}\sigma^2\right)\boldsymbol{\alpha} &= \mathbf{y}. \end{aligned}$$

Since K is positive definite, Q is a convex function and the unique solution given by

$$\boldsymbol{\alpha}^\star = \left(\mathbf{K} + \mathbb{1}\sigma^2\right)^{-1}\mathbf{y}. \tag{A.9}$$

The matrix $(\mathbf{K} + \mathbb{1}\sigma^2)$ is regular for some $\sigma^2 > 0$ and its inverse can be computed by standard techniques in time $\mathcal{O}(n^3)$.

We showed that, from a Bayesian perspective, minimizing the squared error is equivalent to maximizing the likelihood under the assumption that the targets are generated from a smooth function with added Gaussian noise. Notice that the solution of RLSR is equivalent to the mean of a Gaussian process (MacKay, 1998; Rasmussen and Williams, 2006). However, the latter provides a more sophisticated inference due to a more powerful framework.

A.2 Perceptrons

Perceptrons (Rosenblatt, 1958; Duda et al., 2001), also known as single-layer networks (Bishop, 1995), form the elementary unit of artificial neural networks that are made up of several layers of interconnected perceptrons. A perceptron is a linear discriminant function $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$ that classifies an instance $\mathbf{x}$ into a class $y \in \{+1, -1\}$ by applying the signum operator to the decision function $y = \text{sign}(f(\mathbf{x}))$. Given n training instances $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{+1, -1\}$, a perfect separation of the two classes can be expressed as a set of inequalities,

$$\forall_{i=1}^{n} \quad y_i f(\mathbf{x}_i) \geq 0.$$

To achieve this goal, the perceptron incrementally processes one training example at a time, performing an update step whenever an example is misclassified. The perceptron algorithm is initialized with $\mathbf{w}^{(0)} = \mathbf{0}$. When the i-th instance is erroneously classified, that is $y_i f(\mathbf{x}_i) \leq 0$, the weights $\mathbf{w}^{(t)}$ are updated according to the rule

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \eta^{(t)} y_i \mathbf{x}_i, \tag{A.10}$$

with an appropriate sequence $(\eta^{(t)})_{t=1,\ldots,\infty}$ that is also called the learning rate. It has been shown that $\eta^{(t)} = 1/t$ preserves some advantages in terms of convergence rate and the number of updates (Robbins and Monro, 1951; Fukunga, 1990). However, when we apply a constant learning rate $\eta^{(t)} = 1$ for any t, the perceptron algorithm is also guaranteed to converge to a correct solution if such a solution exists. This is the case if the training data is *linearly separable*, that is, if both classes can be perfectly separated by a hyperplane. The number of update steps needed is at most $t \leq (r/\gamma)^2 \, \|\mathbf{w}^{(opt)}\|^2$ (Novikoff, 1962), where the variable r denotes the radius of the smallest hypersphere enclosing the data points and γ is the functional margin (Section A.3).

Reverse engeneering of the perceptron algorithm shows that the weight vector can be written in terms of dual variables $\alpha_i \in \mathbb{N}_0$ counting how many times example i has been used for an update, that is, $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_j \mathbf{x}_i$. Applying this equality to the decision function obtains the dual

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \langle \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i, \mathbf{x} \rangle = \sum_{i=1}^{n} \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle.$$

The inner product can be computed efficiently by a kernel function

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \psi(\mathbf{x}_i), \psi(\mathbf{x}_j) \rangle$$

Table A.1: *Dual Perceptron Algorithm*

Input: Labeled data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$.

```
1 Initialize all αi = 0.
2 repeat
3    for i = 1,...,n
4       if yi ( Σ_{j=1}^{n} αj yj k(xj, xi) ) < 0
5          Increment αi ← αi + 1
6       end
7    end
8 until convergence
```

Output: Trained hypothesis $f(\mathbf{x})$.

that also allows for implicit feature mappings $\psi(\mathbf{x})$. The dual update rule in case of misclassifying the i-th example is then given by incrementing $\alpha_i \leftarrow \alpha_i + 1$, see Algorithm A.1.

The perceptron minimizes the empirical risk with 0/1 loss evading discontinuities in the involved gradients by an incremental procedure. Minimizing the 0/1 frequently leads to ill-posed problems with no unique solution. Due to the incremental nature of the perceptron, the ordering of the training data highly determines the final location of the solution hyperplane. For this reason, several extensions to the perceptron have been proposed such as weight averaging (Brückner and Dilger, 2005) which leads to more stable solutions and provides better generalization abilities (Freund and Shapire, 1999; Gentile, 2001) or incorporating margins (Duda et al., 2001). Collobert and Bengio (2004) indicate the equivalence of regularized margin perceptrons and support vector machines, although their optimization algorithms are different.

A.3 Support Vector Machines

Support vector machines (SVMs) (Boser et al., 1992; Vapnik, 1998) are linear large margin classifiers; their solution is the (unique) hyperplane realizing the maximal margin (the minimal distance) between two classes given the separating hyperplane. This intuitive solution combined with the potential use of kernel functions and sound theoretical results has led to an appealing framework and finally to their success. Support vector learning has strong connections to perceptrons and two-layer neural networks (Cortes and Vap-

nik, 1995; Collobert and Bengio, 2004) and has been extended to other domains such as multi-class classification (Crammer and Singer, 2001), ordinal regression (Herbrich et al., 1999), and function approximation (Smola and Schölkopf, 1998).

Given n instances $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{+1, -1\}$, and a linear hypothesis

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b, \tag{A.11}$$

where, for the sake of completeness, we make explicit use of a threshold b and refer to Poggio et al. (2001) for details on which settings benefit from a threshold and which do not.

Assume for a moment that our n data points are linearly separable. Then the goal of support vector learning is to maximize the minimal distance between the instances and the hyperplane. An indicator is the *functional margin* $\bar{\gamma}$ that is defined as the smallest decision value[1] across the sample

$$\bar{\gamma} = \min_{1 \leq i \leq n} y_i f(\mathbf{x}_i) \quad \Rightarrow \quad \forall_{i=1}^{n} \; y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq \bar{\gamma}. \tag{A.12}$$

The solution vector $\mathbf{w}$ can be scaled arbitrarily. Thus, to ensure uniqueness of the solution, we scale the functional margin by the norm of the weights and derive the *geometrical margin* $\gamma = \bar{\gamma}/\|\mathbf{w}\|$ that is invariant against scaling of $\mathbf{w}$ and equivalent to the Euclidean distance between the hyperplane and the nearest point. To find the maximal margin we need to maximize γ by either maximizing $\bar{\gamma}$ for a fixed weight vector or minimizing $\|\mathbf{w}\|$ for a fixed functional margin. Support vector learning approaches usually follow the latter and minimize the norm of the weight vector subject to the constraints A.12 for fixed $\bar{\gamma} = 1$.

In general, however, the data will not be linearly separable. To deal with arbitrary input problems, nonnegative slack variables $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)^\mathsf{T}$ can be incorporated into the constraints A.12 allowing pointwise relaxations for margin violations. Notice, that incorporating slack variables can be interpreted as using (squared) hinge loss as an upper bound of 0/1-loss. The primal soft-margin support vector optimization problem is defined as follows.

Optimization Problem A.1 *Given labeled examples* $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, *with* $\mathbf{x}_i \in \mathcal{X}$ *and* $y_i \in \{+1, -1\}$, *a feature mapping* $\psi : \mathcal{X} \to \mathcal{F}$, *and constants* $C > 0$ *and* $r \in \{1, 2\}$

$$\min_{\mathbf{w}, \boldsymbol{\xi}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{rn} \sum_{i=1}^{n} \xi_i^r$$

[1] Notice that in case of linearly separable data this value is always positive.

subject to the constraints

$$\begin{aligned} \forall_{i=1}^{n} \quad & y_i \left(\langle \mathbf{w}, \psi(\mathbf{x}_i) \rangle + b \right) \geq 1 - \xi_i \\ \forall_{i=1}^{n} \quad & \xi_i \geq 0. \end{aligned}$$

The parameter C determines the trade-off between margin maximization and error minimization and r penalizes errors linearly or quadratically. The second term in the objective, the sum of the slack variables, upper bounds the empirical error with 0/1-loss since $\xi_i \geq 1$ holds in case of a misclassification. Notice that we can express the slacks equivalently by $\xi_i = (1 - y_i f(\mathbf{x}_i))_+^r$ which is precisely the hinge loss for $r = 1$ and the quadratic loss for $r = 2$ that both upper bound the 0/1 loss (see Figure 2.1).

The constraints of Optimization Problem A.1 can be integrated into the objective by the theorem of *Lagrange* (Boyd and Vandenberghe, 2004). The result is the so-called *Lagrangian* and given by

$$\begin{aligned} Q(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \; & \frac{\|\mathbf{w}\|^2}{2} + \frac{C}{rn} \sum_{i=1}^{n} \xi_i^r - \sum_{i=1}^{n} \beta_i \xi_i \\ & - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\langle \mathbf{w}, \psi(\mathbf{x}_i) \rangle + b \right) - 1 + \xi_i \right) \end{aligned}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^\mathsf{T}$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)^\mathsf{T}$ are called *Lagrange multipliers*. It can be shown that the saddle point of the Lagrangian coincides precisely with the minimum of Optimization Problem A.1 (Boyd and Vandenberghe, 2004).

We describe the derivation of the dual SVM optimization problem for linear penalities in greater detail. For $r = 1$ we derive

$$\frac{\partial Q}{\partial \xi_i} = \frac{C}{n} - \alpha_i - \beta_i \stackrel{!}{=} 0 \quad \Rightarrow \quad \beta_i = \frac{C}{n} - \alpha_i \tag{A.13}$$

which, with $\alpha_i \geq 0$ and $\beta_i \geq 0$, leads to the so-called box-constraints $0 \leq \alpha_i \leq \frac{C}{n}$. Substituting A.13 into the Lagrangian eliminates its direct dependence on the slack variables

$$Q(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i y_i \left(\langle \mathbf{w}, \psi(\mathbf{x}_i) \rangle + b \right) + \sum_{i=1}^{n} \alpha_i. \tag{A.14}$$

Repeating this procedure of differentiating and substituting with respect to the threshold b yields a simplified form that no longer depends explicitly on

the latter (Equation A.15)

$$Q(\mathbf{w}, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i y_i \langle \mathbf{w}, \psi(\mathbf{x}_i)\rangle + \sum_{i=1}^{n} \alpha_i. \tag{A.15}$$

Instead, we capture the influence of variable b implicitly by the constraint

$$\frac{\partial Q}{\partial b} = \sum_{i=1}^{n} \alpha_i y_i \stackrel{!}{=} 0. \tag{A.16}$$

Finally, we remove the dependence of A.15 on the primal weight vector by taking the derivative with respect to the entire vector $\mathbf{w}$. We resolve

$$\frac{\partial Q}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \psi(\mathbf{x}_i) \stackrel{!}{=} 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \psi(\mathbf{x}_i). \tag{A.17}$$

Again, substituting the relation A.17 obtains the dual Lagrangian that depends only on the dual variables, the Lagrange multipliers. Introducing kernel $k(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}')\rangle$, we have

$$\begin{aligned} Q(\boldsymbol{\alpha}) &= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i y_i \langle \mathbf{w}, \psi(\mathbf{x}_i)\rangle + \sum_{i=1}^{n} \alpha_i \\ &= \frac{1}{2}\left(\sum_{i=1}^{n} \alpha_i y_i \psi(\mathbf{x}_i)\right)^2 - \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{n} \alpha_i \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j). \end{aligned} \tag{A.18}$$

We will refer to the equivalent vector notation of Equation A.18 given by

$$Q(\boldsymbol{\alpha}) = \mathbf{1}^\mathsf{T}\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\mathsf{T}\mathbf{Y}\mathbf{K}\mathbf{Y}\boldsymbol{\alpha}, \tag{A.19}$$

with diagonal matrix $\mathbf{Y}$ where $[\mathbf{Y}]_{ii} = y_i$ and gram matrix $\mathbf{K}$ given by $[\mathbf{K}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. Notice that Equation A.17 is in line with the representer theorem. The corresponding dual optimization problem is stated as follows.

Optimization Problem A.2 (L_1 Dual SVM) *Given n labeled examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, with $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \{+1, -1\}$, kernel matrix $\mathbf{K}$ with $[\mathbf{K}]_{ij} = \langle \psi(\mathbf{x}_i), \psi(\mathbf{x}_j)\rangle$, and parameter $C > 0$,*

$$\min_{\boldsymbol{\alpha}} \quad \mathbf{1}^\mathsf{T}\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\mathsf{T}\mathbf{Y}\mathbf{K}\mathbf{Y}\boldsymbol{\alpha}$$

subject to the constraints $\forall_{i=1}^{n}\, 0 \leq \alpha_i \leq \frac{C}{n}$ and $\mathbf{y}^\mathsf{T}\boldsymbol{\alpha} = 0$.

For $r = 2$ the calculations are very similar. We only present the resulting optimization problem with the observation that the derivative with respect to the slack variables does not vanish as before, but can be augmented into the kernel matrix by $\mathbf{K}' = \mathbf{K} + \mathbb{1}\frac{n}{C}$. The interested reader is directed to one of the excellent text books on support vector machines (Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2002; Joachims, 2002).

Optimization Problem A.3 (L_2 Dual SVM) *Given n labeled examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, with $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \{+1, -1\}$, kernel matrix $\mathbf{K}$ with $[\mathbf{K}]_{ij} = \langle \psi(\mathbf{x}_i), \psi(\mathbf{x}_j) \rangle$, and parameter $C > 0$,*

$$\min_{\boldsymbol{\alpha}} \quad \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{Y}\mathbf{K}'\mathbf{Y}\boldsymbol{\alpha}$$

subject to the constraints $\forall_{i=1}^n 0 \leq \alpha_i$ and $\mathbf{y}^T\boldsymbol{\alpha} = 0$, with $\mathbf{K}' = \mathbf{K} + \mathbb{1}\frac{n}{C}$.

It is desirable to minimize optimization Problem A.2 or A.3 instead of Optimization Problem A.1 for $r = 1, 2$, respectively, since the duals exhibit simpler constraints and allow the use of standard quadratic programming techniques (Boyd and Vandenberghe, 2004). For instance, one only has to optimize n dual variables compared to $\dim(\psi)$ primal variables. The overall time needed for the computation of the solution is in $\mathcal{O}(n^3)$. In the case of large sample sizes, decomposition techniques may lead to significant speedups (Platt, 2000; Joachims, 1999b) as well as focusing on linear optimization (Joachims, 2006). Recently, approaches optimizing SVMs in the primal yield excellent results in terms of execution time and accuracy (Keerthi and DeCoste, 2005; Chapelle, 2006; Shalev-Shwartz et al., 2007).

Appendix B

Viterbi Decoding

In this section we prove that for label sequence learning problems, decoding the top scoring output sequence

$$\hat{y} = \operatorname*{argmax}_{\bar{\mathbf{y}} \in \mathcal{Y}(x)} f(x, \bar{y}) \tag{B.1}$$

can be performed by a Viterbi algorithm (Forney, 1973; Schwarz and Chow, 1990) in $\mathcal{O}(T|\Sigma|^2)$, where f is a generalized linear model

$$f(x, y) = \langle \mathbf{w}, \Phi(x, y) \rangle. \tag{B.2}$$

For notational convenience we will use the notation for $\Phi(x, y)$ introduced by Altun et al. (2003b). Given two labels $\sigma, \tau \in \Sigma$, we define

$$\phi^A_{\sigma,\tau}(y_{t-1}, y_t) = [[y_{t-1} = \sigma \wedge y_t = \tau]] \tag{B.3}$$

$$\phi^B_{\sigma,j}(x_t, y_t) = [[y_t = \sigma]]\psi_j(x_t) \tag{B.4}$$

As described in Section 3.2.2, $\psi_j(x)$ extracts characteristics of x (e.g., feature $\psi_{234}(x) = 1$ if token x_t starts with a capital letter and 0 otherwise). We will refer to the vector $\psi(x) = (\ldots, \psi_j(x), \ldots)^\mathsf{T}$ and denote the inner product by means of $k(x, x') = \langle \psi(x), \psi(x') \rangle$.

Analogously to Equation 3.23, Altun et al. (2003b) define the joint feature representation $\Phi(x_i, y_i)$ of the i-th sequence as the sum of all feature vectors $\Phi(x_i, y_i|t) = (\ldots, \phi^A_{\sigma,\tau}(y_{i,t-1}, y_{i,t}), \ldots, \phi^B_{\sigma,j}(x_{i,t}, y_{i,t}), \ldots)^\mathsf{T}$ extracted at time t

$$\Phi(x_i, y_i) = \sum_{t=1}^{T_i} \Phi(x_i, y_i|t). \tag{B.5}$$

The feature map in Equation B.5 gives rise to the following inner product in input-output space that decomposes into a *label-label* and a *label-observation*

part,

$$\begin{aligned}
&\langle \Phi(x, y), \Phi(x', y') \rangle \\
&\quad = \sum_{s,t} \sum_{\sigma,\tau} [[y_{s-1} = \sigma \wedge y_s = \tau]] \cdot [[y'_{t-1} = \sigma \wedge y'_t = \tau]] \\
&\qquad + \sum_{s,t} \sum_{\sigma} [[y_s = \sigma]] \psi(x_s) \cdot [[y'_t = \sigma]] \psi(x'_t) \\
&\quad = \sum_{s,t} [[y_{s-1} = y'_{t-1} \wedge y_s = y'_t]] + \sum_{s,t} [[y_s = y'_t]] k(x_s, x'_t). \qquad \text{(B.6)}
\end{aligned}$$

The Viterbi algorithm uses dynamic programming to maintain a trellis in which nodes correspond to hidden states versus times. Each entry stores the score of the most probable path leading to that node at a certain time, see Figure B.1. Once the computation reaches the end of the sequence, it backtracks the most likely path through the trellis and returns the highest scoring label sequence that generates the input sequence. The following Proposition B.1 shows that the argument of the maximum can be computed by a Viterbi algorithm.

Proposition B.1 *Given n input-output pairs of sequences of length T_i for $1 \leq i \leq n$, let Σ denote the output alphabet with $|\Sigma| < \infty$. Let f be defined as*

$$f(x, y) = \sum_{i=1}^{n} \sum_{\substack{\bar{y} \in \mathcal{Y}(x_i) \\ \bar{y} \neq y_i}} \alpha_i(\bar{y}) \left(\langle \Phi(x_i, y_i), \Phi(x, y) \rangle - \langle \Phi(x_i, \bar{y}), \Phi(x, y) \rangle \right),$$

with the joint feature map $\Phi(x, y)$ as in Equation B.5. Then for all $\alpha_i(\bar{y}) \geq 0$ and any $x \in \mathcal{X}$,

$$\hat{y} = \operatorname*{argmax}_{\bar{y} \in \mathcal{Y}(x)} f(x, \bar{y})$$

can be computed with a Viterbi algorithm in time $\mathcal{O}(T|\Sigma|^2)$.

Proof The model f has the form

$$\begin{aligned}
f(x, y) &= \sum_{i=1}^{n} \sum_{\substack{\bar{y} \in \mathcal{Y}(x_i) \\ \bar{y} \neq y_i}} \alpha_i(\bar{y}) \left(\langle \Phi(x_i, y_i), \Phi(x, y) \rangle - \langle \Phi(x_i, \bar{y}), \Phi(x, y) \rangle \right) \\
&= \sum_{i=1}^{n} \sum_{\substack{\bar{y} \in \mathcal{Y}(x_i) \\ \bar{y} \neq y_i}} \alpha_i(\bar{y}) \left(\sum_{s,t} \left([[y_{i,s} = y_t]] - [[\bar{y}_s = y_t]] \right) k(x_{i,s}, x_t) \right. \\
&\qquad \left. + \sum_{s,t} [[y_{i,s-1} = y_{t-1} \wedge y_{i,s} = y_t]] - [[\bar{y}_{s-1} = y_{t-1} \wedge \bar{y}_s = y_t]] \right).
\end{aligned}$$

We make the dependency on labels $\sigma, \tau \in \Sigma$ explicit by summing over all transitions and observation states

$$f(\mathcal{x}, y) = \sum_{\sigma,\tau\in\Sigma} \sum_{i} \sum_{\substack{\bar{y}\in\mathcal{Y}(x_i)\\ \bar{y}\neq y_i}} \alpha_i(\bar{y}) \Big(\sum_{s,t} \left([[y_{i,s} = \sigma]] - [[\bar{y}_s = \sigma]]\right) [[y_t = \tau]] k(x_{i,s}, x_t)$$

$$+ \sum_{s,t} \Big([[y_{i,s-1} = \sigma \wedge y_{i,s} = \tau]] - [[\bar{y}_{s-1} = \sigma \wedge \bar{y}_s = \tau]] \Big) [[y_{t-1} = \sigma \wedge y_t = \tau]] \Big) .$$

The transition scores from label σ to label τ are now given by

$$a(\sigma, \tau) = \sum_{i=1}^{n} \sum_{\substack{\bar{y}\in\mathcal{Y}(x_i)\\ \bar{y}\neq y_i}} \alpha_i(\bar{y}) \Big(\sum_{t=1}^{T_i} [[y_{i,t-1} = \sigma \wedge y_{i,t} = \tau]] - [[\bar{y}_{t-1} = \sigma \wedge \bar{y}_t = \tau]] \Big)$$

and observation scores for label $y_s = \sigma$ and observation x_s by

$$b(\sigma, x) = \sum_{i=1}^{n} \sum_{t=1}^{T_i} \sum_{\substack{\bar{y}\in\mathcal{Y}(x_i)\\ \bar{y}\neq y_i}} \alpha_i(\bar{y}) \left([[y_{i,t} = \sigma]] - [[\bar{y}_t = \sigma]]\right) k(x_{i,t}, x).$$

The hypothesis $f(\mathcal{x}, y)$ can be rewritten in terms of transition scores $a(\sigma, \tau)$ and observation scores $b(\sigma, x)$

$$f(\mathcal{x}, y) = \underbrace{\sum_{\sigma,\tau\in\Sigma} a(\sigma, \tau) \sum_{s=1}^{T} [[y_{s-1} = \sigma \wedge y_s = \tau]]}_{=: f_a(\mathcal{x}, y)} + \underbrace{\sum_{s=1}^{T} \sum_{\sigma\in\Sigma} [[y_s = \sigma]] b(\sigma, x_s)}_{=: f_b(\mathcal{x}, y)} .$$

where f_a weights the occurences of neighboring labels in y by corresponding scores of the model and f_b determines how well observations x_s fit to their labels y_s given the model. To decode the top scoring sequence we define

$$\delta_t(\sigma) = \max_{y_1,\ldots,y_{t-1}} f(\mathcal{x}, y_1, \ldots, y_{t-1}, y_t = \sigma), \tag{B.7}$$

that is, $\delta_t(\sigma)$ denotes the top scoring partial sequence up to position $t-1$ where $y_t = \sigma$. We first show by induction that

$$\delta_{t+1}(\sigma) = \max_{\tau\in\Sigma} \left[\delta_t(\tau) + a(\tau, \sigma)\right] + b(\sigma, x_{t+1}) \tag{B.8}$$

holds. The initialization is simply given by

$$\begin{aligned} \delta_0(\sigma) &= 0, \quad \forall \sigma \in \Sigma \\ \delta_1(\sigma) &= \max_{\tau\in\Sigma} \left[\delta_t(\tau) + a(\tau, \sigma)\right] + b(\sigma, x_{t+1}) \\ &= a(\epsilon, \sigma) + b(\sigma, x_1). \end{aligned}$$

The recursion step is given for $2 \leq t \leq T$ by

$$\begin{aligned}
\delta_t(\sigma) &= \max_{y_1,\ldots,y_{t-1}} f(\mathcal{x}, y_1, \ldots, y_{t-1}, y_t = \sigma) \\
&= \max_{y_1,\ldots,y_{t-1}} \sum_{\tau,\bar{\tau}\in\mathcal{Y}} a(\tau,\bar{\tau}) \sum_{s=2}^{t-1} [[y_{s-1} = \tau \wedge y_s = \bar{\tau}]] \\
&\qquad + \sum_{\tau\in\Sigma} a(\tau,\sigma)[[y_{t-1} = \tau \wedge y_t = \sigma]] \\
&\qquad + \sum_{s=1}^{t-1} \sum_{\tau\in\Sigma} [[y_s = \tau]] b(\tau, x_s) + [[y_t = \sigma]] b(\sigma, x_t) \\
&= \max_{\sigma^\star} \max_{y_1,\ldots,y_{t-2}} \sum_{\tau,\bar{\tau}\in\mathcal{Y}} a(\tau,\bar{\tau}) \sum_{s=2}^{t-2} [[y_{s-1} = \tau \wedge y_s = \bar{\tau}]] \\
&\qquad + \sum_{\tau\in\Sigma} a(\tau,\sigma^\star)[[y_{t-2} = \tau \wedge y_{t-1} = \sigma^\star]] \\
&\qquad + a(\sigma^\star,\sigma)[[y^{t-1} = \sigma^\star \wedge y^t = \sigma]] \\
&\qquad + \sum_{s=1}^{t-2} \sum_{\tau\in\Sigma} [[y_s = \tau]] b(\tau, x_s) + b(\sigma^\star, x_{t-1}) + b(\sigma, x_t) \\
&= \max_{\sigma^\star} \left[\max_{y_1,\ldots,y_{t-2}} f(\mathcal{x}, y_1, \ldots, y_{t-2}, y_{t-1} = \sigma^\star) + a(\sigma^\star,\sigma) \right] + b(\sigma, x_t) \\
&= \max_{\sigma^\star} \left[\delta_{t-1}(\sigma^\star) + a(\sigma^\star,\sigma) \right] + b(\sigma, x_t).
\end{aligned}$$

Thus, the top scoring sequence has the score

$$\max f(\mathcal{x}, y) = \max_{\sigma\in\Sigma} \delta_T(\sigma).$$

We only sketch the extension to the argument of the maximum since it is analoguous to the regular Viterbi algorithm. We introduce path variables $\varphi_t(\sigma)$ that are initialized by $\varphi_1(\sigma) = \epsilon$ for all $\sigma \in \Sigma$. The sequence $\varphi_t(\sigma)$ is then defined recursively for $2 \leq t \leq T$ by

$$\varphi_t(\sigma) = \operatorname*{argmax}_{\sigma^\star\in\Sigma} \left[\delta_{t-1}(\sigma^\star) + a(\sigma^\star,\sigma) \right].$$

Once the $\delta_t(\sigma)$ of Proposition B.1 are fixed, the optimal label sequence can be found by backtracking

$$\begin{aligned}
y_T^\star &= \operatorname*{argmax}_{\sigma\in\Sigma} \delta_T(\sigma) \\
y_t^\star &= \varphi_{t+1}(y_{t+1}^\star) \quad \text{for} \quad t = T-1, \ldots, 1.
\end{aligned}$$

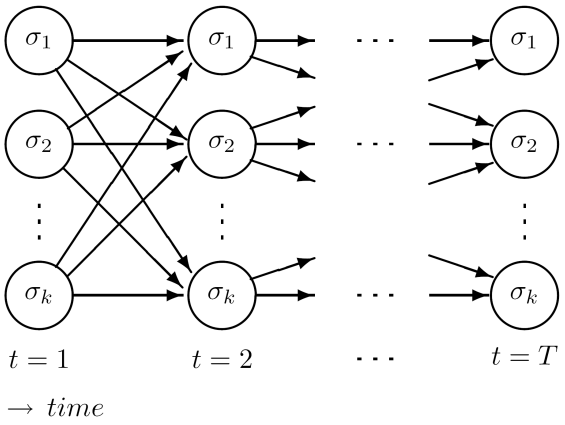


Figure B.1: *Visualization of a trellis over the alphabet* $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$.

Given the transition matrix $[\mathbf{A}]_{\sigma,\tau} = a(\sigma, \tau)$ and the observation matrix $[\mathbf{B}_x]_{\sigma,t} = b(\sigma, x_t)$ for input x, the computation of δ and φ for a fixed t and $\sigma \in \Sigma$ involves visiting $|\Sigma|$ predecessors; thus, for a sequence of length T the time needed is in $\mathcal{O}(T|\Sigma|^2)$. This concludes the proof. □

In V-view learning, the joint decision function used for inference is given by

$$f(x, y) = f^1(x, y) + f^2(x, y) + \ldots + f^V(x, y) \tag{B.9}$$

We show in Proposition B.2 that Proposition B.1 also holds for the joint decision function.

Proposition B.2 *The prediction* $\hat{y}$ *of the joint decision function*

$$\hat{y} = \operatorname*{argmax}_{\bar{y} \in \mathcal{Y}(x)} f(x, \bar{y}) = \operatorname*{argmax}_{\bar{y} \in \mathcal{Y}(x)} \left[f^1(x, \bar{y}) + \ldots + f^V(x, \bar{y}) \right] \tag{B.10}$$

with $f^v(x, y) = \langle \mathbf{w}^v, \Phi^v(x, y) \rangle$ *for* $v = 1, \ldots, V$*, where* Φ^v *is defined in Equation B.5, can be computed by a Viterbi decoding.*

Proof According to Proposition B.1 all views $v = 1, \ldots, V$ provide transition scores from label σ to label τ,

$$a^v(\sigma, \tau) = \sum_{i=1}^{n} \sum_{\substack{\bar{y} \in \mathcal{Y}(x_i) \\ \bar{y} \neq y_i}} \alpha_i^v(\bar{y}) \underbrace{\left(\sum_{t=1}^{T_i} [[y_{i,t-1} = \sigma \wedge y_{i,t} = \tau]] - [[\bar{y}_{t-1} = \sigma \wedge \bar{y}_t = \tau]] \right)}_{=: const(i, \sigma, \tau, \bar{y})}$$

and observation scores

$$b^v(\sigma, x) = \sum_{i=1}^{n} \sum_{t=1}^{T_i} \sum_{\substack{\bar{y} \in \mathcal{Y}(x_i) \\ \bar{y} \neq y_i}} \alpha_i^v(\bar{y}) \underbrace{\left([[y_{i,t} = \sigma]] - [[\bar{y}_t = \sigma]] \right)}_{=: const(i,t,\sigma,\bar{y})} k^v(x_{i,t}, x),$$

for observing x_s while in state $y_s = \sigma$. The joint transition scores are defined over all V views given by

$$\begin{aligned} a(\sigma, \tau) =& a^1(\sigma, \tau) + \ldots + a^V(\sigma, \tau) \\ =& \sum_{i=1}^{n} \sum_{\substack{\bar{y} \in \mathcal{Y}(x_i) \\ \bar{y} \neq y_i}} \alpha_i^1(\bar{y})\, const(i, \sigma, \tau, \bar{y}) + \sum_{i=1}^{n} \sum_{\substack{\bar{y} \in \mathcal{Y}(x_i) \\ \bar{y} \neq y_i}} \alpha_i^2(\bar{y})\, const(i, \sigma, \tau, \bar{y}) \\ & + \ldots + \sum_{i=1}^{n} \sum_{\substack{\bar{y} \in \mathcal{Y}(x_i) \\ \bar{y} \neq y_i}} \alpha_i^V(\bar{y})\, const(i, \sigma, \tau, \bar{y}). \end{aligned}$$

Rearranging the terms leads to

$$a(\sigma, \tau) = \sum_{i=1}^{n} \sum_{\substack{\bar{y} \in \mathcal{Y}(x_i) \\ \bar{y} \neq y_i}} \left(\alpha_i^1(\bar{y}) + \ldots + \alpha_i^V(\bar{y}) \right) const(i, \sigma, \tau, \bar{y}).$$

Similarly, the joint observation scores have the form

$$b(\sigma, x) = \sum_{i,t} \sum_{\substack{\bar{y} \in \mathcal{Y}(x_i) \\ \bar{y} \neq y_i}} \left(\alpha_i^1(\bar{y}) k^1(x_{i,t}, x) + \ldots + \alpha_i^V(\bar{y}) k^V(x_{i,t}, x) \right) const(i, t, \sigma, \bar{y}).$$

In terms of the joint score functions the joint hypothesis $f(\mathcal{x}, y)$ can be written in the well known form

$$f(\mathcal{x}, y) = \sum_{\sigma, \tau \in \Sigma} a(\sigma, \tau) \sum_{s=1}^{T} [[y_{s-1} = \sigma \wedge y_s = \tau]] + \sum_{s=1}^{T} \sum_{\sigma \in \Sigma} [[y_s = \sigma]] b(\sigma, x_s).$$

Applying Proposition B.1 proves the claim. □

Appendix C

Cocke-Kasami-Younger Parsing

In this chapter, we show that the top scoring parse tree $\hat{y}$ for a given input sentence $\mathcal{x}$ given by

$$\hat{y} = \underset{\bar{y} \in \mathcal{Y}(\mathcal{x})}{\operatorname{argmax}} f(\mathcal{x}, \bar{y}) \tag{C.1}$$

can be decoded with a Cocke-Kasami-Younger (CKY) algorithm with respect to a weighted context-free grammar $\mathcal{G}$ in Chomsky normal form (Kasami, 1965; Younger, 1967). In the following we make use of the generalized linear model

$$f(\mathcal{x}, y) = \langle \mathbf{w}, \Phi(\mathcal{x}, y) \rangle \tag{C.2}$$

and the joint feature mapping $\Phi(\mathcal{x}, y)$ defined in Equation 3.29. Note that all results are easily generalizable to feature mappings that include complex (local) features. Before we present the main result of this chapter let us briefly introduce some common notation that may help to understand the further derivations.

Definition C.1 *Let $\mathcal{G} = (\mathcal{N}, \mathcal{T}, \Sigma, S)$ be a context-free grammar as in Definition 3.3. For any production $(A \to \alpha) \in \Sigma$ and any $u, v \in (\mathcal{N} \times \mathcal{T})^*$ we write $uAv \Rightarrow_{\mathcal{G}} u\alpha v$ to indicate that $u\alpha v$ can be directly derived from uAv given $\mathcal{G}$. If $\alpha_1, \ldots, \alpha_m \in (\mathcal{N} \times \mathcal{T})^*$ and*

$$\alpha_1 \Rightarrow_{\mathcal{G}} \alpha_2, \quad \alpha_2 \Rightarrow_{\mathcal{G}} \alpha_3, \quad \ldots, \quad \alpha_{m-1} \Rightarrow_{\mathcal{G}} \alpha_m,$$

holds, we write $\alpha_1 \Rightarrow_{\mathcal{G}}^* \alpha_m$, where $\Rightarrow_{\mathcal{G}}^*$ denotes the reflexive and transitive closure of $\Rightarrow_{\mathcal{G}}$.

We now show the main result of this chapter.

Proposition C.1 *Let $\mathcal{G} = (\mathcal{N}, \mathcal{T}, \Sigma, S)$ be a PCFG in Chomsky normal form defined in Definitions 3.4 and 3.5 that generates sentences $\boldsymbol{x} \in \mathcal{X}$ with $\boldsymbol{x} = x_1, \ldots, x_T$ and parse tree $\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})$. Moreover, let f be defined as in Equation C.2 and Φ as in Equation 3.29. Then for any $\boldsymbol{x} \in \mathcal{X}$,*

$$\hat{y} = \operatorname*{argmax}_{\bar{y} \in \mathcal{Y}(\boldsymbol{x})} f(\boldsymbol{x}, \bar{y}) = \operatorname*{argmax}_{\bar{y} \in \mathcal{Y}(\boldsymbol{x})} \langle \mathbf{w}, \Phi(\boldsymbol{x}, \bar{y}) \rangle,$$

can be computed with a CKY algorithm in time $\mathcal{O}(|\Sigma| T^3)$ where T is the length of $\boldsymbol{x}$.

Proof Let $y_{[i,j]} \in \Sigma \cup \{\emptyset\}$ denote the (non-terminal) root of the parse $\boldsymbol{y}_{[i,j]}$ that generates the subsequence $\boldsymbol{x}_{[i,j]} = x_i, \ldots, x_j$ with $1 \leq i \leq j \leq T$ or $\emptyset$ if such a parse does not exist. We denote by $\boldsymbol{y}^{sub}_{[i,j]}$ the subtrees of $\boldsymbol{y}_{[i,j]}$ below root $y_{[i,j]}$ as shown in Figure C.1. We define variable $\delta_{i,j}(A)$ that gives us the score of joining optimal subtrees $\boldsymbol{y}^{sub}_{[i,j]}$ by a production with head $A \in \mathcal{N}$

$$\delta_{i,j}(A) = \max_{\boldsymbol{y}^{sub}_{[i,j]} \in \mathcal{Y}(\boldsymbol{x}_{[i,j]})} f(\boldsymbol{x}, \boldsymbol{y}^{sub}_{[i,j]}, y_{[i,j]} = A). \tag{C.3}$$

We show that Equation C.3 requires the existence of two optimal parses $B \Rightarrow^*_{\mathcal{G}} x_i, \ldots, x_k$ and $C \Rightarrow^*_{\mathcal{G}} x_{k+1}, \ldots, x_j$ with respect to k and production rule $(A \to BC) \in \Sigma$. If there is no rule $A \to BC$ or the parse rooted in either B or C does not exist, then the local probability $P(A \to BC) = 0$. In this case we define $\ln P(A \to BC) = \ln 0 := -\infty$ and thus $\delta_{i,j}(A) = -\infty$. We now show that for every pair (i, j), $i \leq j$, either

$$\delta_{i,i}(A) = \langle \mathbf{w}, \Phi(\boldsymbol{x}, A \to x_i) \rangle,$$

for $i = j$ or

$$\delta_{i,j}(A) = \max_{\substack{i \leq k < j \\ B,C:(A \to BC) \in \Sigma}} \left[\langle \mathbf{w}, \Phi(\boldsymbol{x}, A \to BC) \rangle + \delta_{i,k}(B) + \delta_{k+1,j}(C) \right],$$

for $i < j$ holds.

Initialization: For $i = j$ we consider the parse below $y_{[i,i]} = A$ that generates token x_i, i.e., $A \Rightarrow^*_{\mathcal{G}} x_i$. This holds if and only if Σ contains a production $A \to x_i$ since by Definition 3.5 only unary rules are allowed to produce terminals. Thus, we initialize

$$\delta_{i,i}(A) = \langle \mathbf{w}, \Phi(\boldsymbol{x}, A \to x_i) \rangle, \quad \forall 1 \leq i \leq T. \tag{C.4}$$

Note that if there is no such rule $(A \to x_i) \in \Sigma$, or equivalently $P(A \to x_i) = 0$, the inner product in Equation C.4 is not well defined and we define $\delta_{i,i}(A) = -\infty$ instead.

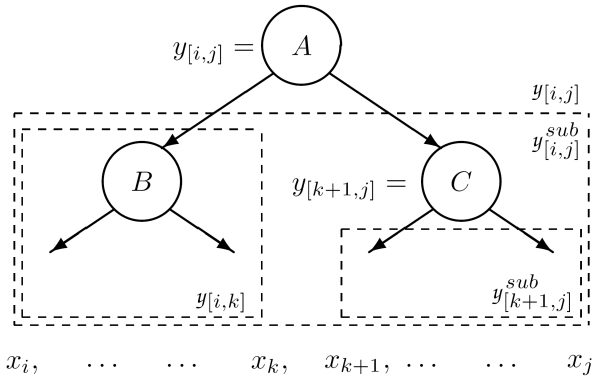


Figure C.1: *Illustration of the notation used in Proposition C.1*

Recursion: To produce the subsequence $x_i, \ldots, x_j$ for $1 \leq i < j \leq T$ we have to apply binary rules (see again Definition 3.5) and hence $y_{[i,j]}$ has to be the head of a binary rule. We resolve

$$
\begin{aligned}
\delta_{i,j}(A) &= \max_{y^{sub}_{[i,j]} \in \mathcal{Y}(x_{[i,j]})} f(x, y^{sub}_{[i,j]}, y_{[i,j]} = A) \\
&= \max_{y^{sub}_{[i,j]} \in \mathcal{Y}(x_{[i,j]})} \langle \mathbf{w}, \Phi(x, y^{sub}_{[i,j]}, y_{[i,j]} = A) \rangle \\
&= \max_{\substack{i \leq k < j \\ y^{sub}_{[i,j]} \in \mathcal{Y}(x_{[i,j]})}} \langle \mathbf{w}, \Phi(x, A \to y_{[i,k]} y_{[k+1,j]}) \rangle + \langle \mathbf{w}, \Phi(x, y^{sub}_{[i,j]}) \rangle \\
&= \max_{\substack{i \leq k < j \\ y_{[i,k]}, y_{[k+1,j]} \in \mathcal{N}}} \Big[\langle \mathbf{w}, \Phi(x, A \to y_{[i,k]} y_{[k+1,j]}) \rangle \\
&\quad + \max_{y_{[i,k]} \in \mathcal{Y}(x_{[i,k]})} \langle \mathbf{w}, \Phi(x, y_{[i,k]}) \rangle + \max_{y_{[k+1,j]} \in \mathcal{Y}(x_{[k+1,j]})} \langle \mathbf{w}, \Phi(x, y_{[k+1,j]}) \rangle \Big] \\
&= \max_{\substack{i \leq k < j \\ B, C : (A \to BC) \in \Sigma}} \Big[\langle \mathbf{w}, \Phi(x, A \to BC) \rangle + \max_{y_{[i,k]} \in \mathcal{Y}(x_{[i,k]})} f(x, y_{[i,k]}, y_{[i,k]} = B) \\
&\quad + \max_{y_{[k+1,j]} \in \mathcal{Y}(x_{[k+1,j]})} f(x, y_{[k+1,j]}, y_{[k+1,j]} = C) \Big] \\
&= \max_{\substack{i \leq k < j \\ B, C : (A \to BC) \in \Sigma}} \left[\langle \mathbf{w}, \Phi(x, A \to BC) \rangle + \delta_{i,k}(B) + \delta_{k+1,j}(C) \right].
\end{aligned}
$$

Termination: Finally, we consider the case $i = 1$ and $j = T$, that is the generation of $x_1, \ldots, x_T$ which is precisely the input x. By definition $S \in \mathcal{N}$

is the only start symbol in $\mathcal{G}$, thus,

$$\begin{aligned}\delta_{1,T}(S) &= \max_{y^{sub}_{[1,T]} \in \mathcal{Y}(\mathcal{x})} f(\mathcal{x}, y^{sub}_{[1,T]}, y_{[1,T]} = S) \\ &= \max_{y \in \mathcal{Y}(\mathcal{x})} f(\mathcal{x}, y)\end{aligned}$$

Taking the argument of the maximum decodes the top scoring parse for subsequence $x_i, \ldots, x_j$ with head A (Equation C.5).

$$\varphi_{i,j}(A) = \underset{\substack{i \leq k < j \\ B,C:(A \to BC) \in \Sigma}}{\operatorname{argmax}} \left[\langle \mathbf{w}, \Phi(\mathcal{x}, A \to BC) \rangle + \delta_{i,k}(B) + \delta_{k+1,j}(C)\right] \tag{C.5}$$

Thus, we can compute the top scoring parse tree for a given input sentence according to

$$\underset{\bar{y} \in \mathcal{Y}(\mathcal{x})}{\operatorname{argmax}} f(\mathcal{x}, \bar{y}) = \varphi_{1,T}(S).$$

Note that if $\delta_{1,T}(S) = -\infty$ or equivalently $\varphi_{1,T}(S) = \emptyset$ a valid path $S \Rightarrow^*_{\mathcal{G}} \mathcal{x}$ does not exist.

The variables $\delta_{i,j}$ have to be computed for all pairs $1 \leq i, j \leq n$ where for each pair (i, j) an appropriate $i \leq k < j$ has to be found. Moreover, every $\delta_{i,j}$ has to be evaluated for every $\sigma \in \Sigma$ and therefore the overall time needed for the computation of the argmax is $\mathcal{O}(|\Sigma| T^3)$. □

One might think of variable y as a programming table, also known as a chart. Figure C.2 shows the solution of the CKY algorithm for the sentence `"Curiosity kills the cat"` where $l = 4 - j + 1$. Compare this solution with the corresponding parse tree in Figure 3.4.

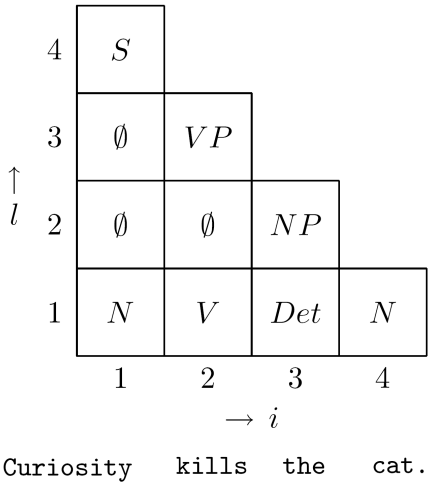


Figure C.2: *Chart displaying the solution of the CKY algorithm.*

Abbrevations

Sets and operations on sets:

$\mathbb{N}$	set of all natural numbers, $1, 2, 3, \ldots$
$\mathbb{R}, \mathbb{R}_0^+$	set of all real numbers and set of nonnegative real numbers
$A \cup B, A \cap B, A - B$	set union, intersection, and difference
$\lvert A \rvert$	number of elements in the set A

Matrices, vectors, and scalars:

$\mathbf{K}, \mathbf{A}, \ldots$	matrices
$\boldsymbol{\alpha}, \mathbf{x}, \ldots$	column vectors
$\mathbf{A}^\mathsf{T}, \mathbf{x}^\mathsf{T}$	transpose of a matrix, row vector
$\eta, \lambda, y, \ldots$	scalars
$\lVert \mathbf{x} \rVert, \lvert a \rvert$	norm of vector $\mathbf{x}$ and absolute value of a
$\langle \mathbf{x}, \bar{\mathbf{x}} \rangle$, $\mathbf{x}^\mathsf{T} \bar{\mathbf{x}}$	inner product of $\mathbf{x}$ and $\bar{\mathbf{x}}$
$\mathbf{0}, \mathbf{1}$	vectors of all zeros and ones
$\mathbb{1}$	identity matrix
$[\mathbf{A}]_{ij}$	element in i-th row and j-th column of $\mathbf{A}$

Complex variables:

$\mathcal{x}, y, z$	structured input and output variables
$\mathcal{X}$	domain of complex inputs
$\mathcal{Y}, \mathcal{Y}(\mathcal{x})$	structured output space, induced output space for input $\mathcal{x}$
$\Phi(\mathcal{x}, y)$	joint feature mapping

Acknowledgment

Foremost, I would like to thank Tobias Scheffer for his tireless pursuit of excellence in any facet of academic life, his invaluable guidance, and for giving me the opportunity to work in a stimulating research context. I would also like to thank Thomas Gärtner and Alex Zien for inspiring discussions, thought-provoking questions, and of course for pushing our ideas forward together. I enjoyed working with you a lot!

I am profoundly grateful to Michael Brückner, Peter Haider, Thoralf Klein, and Christina Limbird for thoroughly proofreading the final version of this thesis. I also deeply appreciate the support of all current and previous members of Tobias' research group at Humboldt-Universität zu Berlin and later at Max-Planck-Institute in Saarbrücken, including Christoph Büscher, Laura Dietz, Isabel Drost, Marcel Gestewitz, Thoralf Klein, Sascha Schulz, and Peter Siemen. Special thanks also to Steffen Bickel and Michael Brückner for sharing the office and their thoughts with me. To Uwe Dick and Peter Haider a big thank you for letting me win a two-digit number of shots in our tabletop soccer rounds: keep it up guys!

My biggest thanks go to my family and all my friends who had absolutely nothing to do with this thesis but nonetheless affectionately listened to my ups and downs at many unlikely hours. To Poldi, thank you for giving me such a great time next to you.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass

(i) ich die vorliegende Dissertationsschrift selbstständig und ohne unerlaubte Hilfe verfasst habe;

(ii) ich mich nicht bereits anderwärts um einen Doktorgrad beworben habe oder einen solchen besitze;

(iii) mir die Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät II der Humboldt-Universität zu Berlin gemäß des amtlichen Mitteilungsblattes Nr. 34/2006 bekannt ist.

Ulf Brefeld
Berlin, 27. Juli 2007